AN210781

带有蓝牙低功耗（BLE）连接的 PSoC 6 MCU 入门

作者: Srinivas Nudurupati, Jim Trudeau

相关部件系列: CY8C63BL

软件版本: PSoC® Creator™ 4.2 或更高版本

相关的 应用笔记和 代码示例：点击这里.

更多代码示例？我们倾听到了你的声音。

要查看持续增长的成百上千的 PSoC 代码示例，请访问代码示例网页。您也可以 在 这里浏览 PSoC 视频库 。

AN210781 介绍采用蓝牙低功耗（BLE）连接的 PSoC 6 MCU，这是一款基于双核 Arm®Cortex®-M4 和 Cortex-M0 +的可编程片上系统，集成了 BLE 无线电系统、最新一代 CapSense®技术和安全功能的主机。本应用笔记可帮助您了解具有 BLE 架构和开发工具的 PSoC 6 MCU，并向您展示如何使用 PSoC Creator 创建您的第一个项目，将项目导出到第三方集成开发环境（IDE），并继续固件开发。 它还可以引导您获得更多的在线可用资源，以加速您学习配备 BLE 的 PSoC 6 MCU。

目录

1 简介 2
1.1 先决条件....3
2 带 BLE 连接 资源的 PSoC 6 MCU 4
2.1 PSoC Creator....4
2.2 PSoC Creator 帮助 5
2.3 代码示例....6
3 PSoC 6 MCU BLE 连接器件特性....7
4 具备 BLE 连接开发生态系统的 PSoC 6 MCU 9
4.1 配置 BLE 连接先锋套件的 PSoC 63....9
4.2 固件/应用开发 9
4.3 外设驱动库 9
4.4 PSoC Creator....10
4.5 支持其它 IDE....10
4.6 RTOS 支持....14
4.7 调试....14
4.8 CySmart 主机仿真工具和移动应用程序....14
5 PSoC 6 MCU BLE 连接性开发设置 15
6 我的第一款采用 BLE 的 PSoC 6 MCU 设计 17
6.1 使用以下指令 17
6.2 准备工作....17
6.3 关于设计....18
6.4 Part 1:从零开始创建新项目 19
6.5 Part 2: 进行设计 23
6.6 Part 3:生成源代码 40
6.7 Part 4:编写固件....43
6.8 Part 5: 编译项目并对设备编程 52
6.9 Part 6: 测试您的设计....54
7 总结 60
8 相关应用笔记和代码示例 60
Appendix A. 赛普拉斯专业术语 62
Appendix B. BLE 协议 63
B.1 概述....63
B.2 物理层 (PHY) 63
B.3 链路层 (LL)....64
B.4 主机控制接口 (HCI)....64
B.5 逻辑链路控制及适配协议 (L2CAP)....65
B.6 安全管理器(SM) 65
B.7 属性协议(ATT) 65
B.8 通用属性配置文件(GATT) 68
B.9 通用访问配置文件 (GAP) 69
Appendix C. PSoC 6 MCU BLE 器件特性 72
C.1 系统资源 72
C.2 安全引导 74
C.3 可编程数字外设....74
C.4 可编程模拟外设....77
C.5 可编程 GPIO....78
Appendix D. 赛普拉斯物联网开发工具 79
D.1 带 BLE Pioneer 先锋套件的 PSoC 63 79

D.2 CySmart 主机仿真工具.....79
D.3 CySmart 移动应用程序.....81
修订记录.....83
销售、解决方案以及法律信息.....84
产品 84
PSoC®解决方案.....84
赛普拉斯开发者社区.....84
技术支持.....84

1 简介

具有 BLE 连接功能的 PSoC 6 MCU（以下简称 PSoC 6 BLE）是赛普拉斯专为可穿戴设备和物联网（IoT）产品设计的超低功耗 PSoC 器件。 它为今天的“永远在线”应用建立了新的低功耗标准。 赛普拉斯 PSoC 6 BLE 器件是一款可编程嵌入式片上系统，可将所有以下产品集成在一个芯片上：

- 双核 Arm®Cortex®-M4 和 Cortex-M0 +微控制器
- BLE 4.2 无线子系统
- 可编程模拟和数字外设
- 高达 1 MB 的闪存和 288 KB 的 SRAM
- 第四代 CapSense®技术

PSoC 6 BLE 适用于各种功耗敏感的连接应用，例如：

- 智能手表和健身追踪器
- 连接的医疗设备
- 智能家居传感器和控制器
- 智能家电
- 游戏控制器
- 运动，智能手机和虚拟现实（VR）配件
- 工业传感器节点
- 工业逻辑控制器
- 高级遥控器

PSoC 6 BLE 为 MCU 和 BLE 无线的组合提供了合算的和所需空间小的替代方案。可编程模拟和数字子系统使用 PSoC Creator（用于开发 PSoC 6 BLE 应用的基于原理图的设计工具）实现了设计的灵活性和动态微调。要开发 BLE 应用，您不需要了解复杂的 BLE 协议栈的相关知识。赛普拉斯在 PSoC Creator 中提供了一个易于配置，免费的基于 GUI 的 BLE 组件，该组件简化了协议的复杂性。

BLE 是由 Bluetooth Special Interest Group（SIG）定义的超低功耗无线标准，用于短距离通信。 PSoC 6 BLE 将 BLE 4.2 无线和免版税协议栈集成在一起，增强了安全性，隐私性和吞吐量，符合 BLE 5.0 规范。

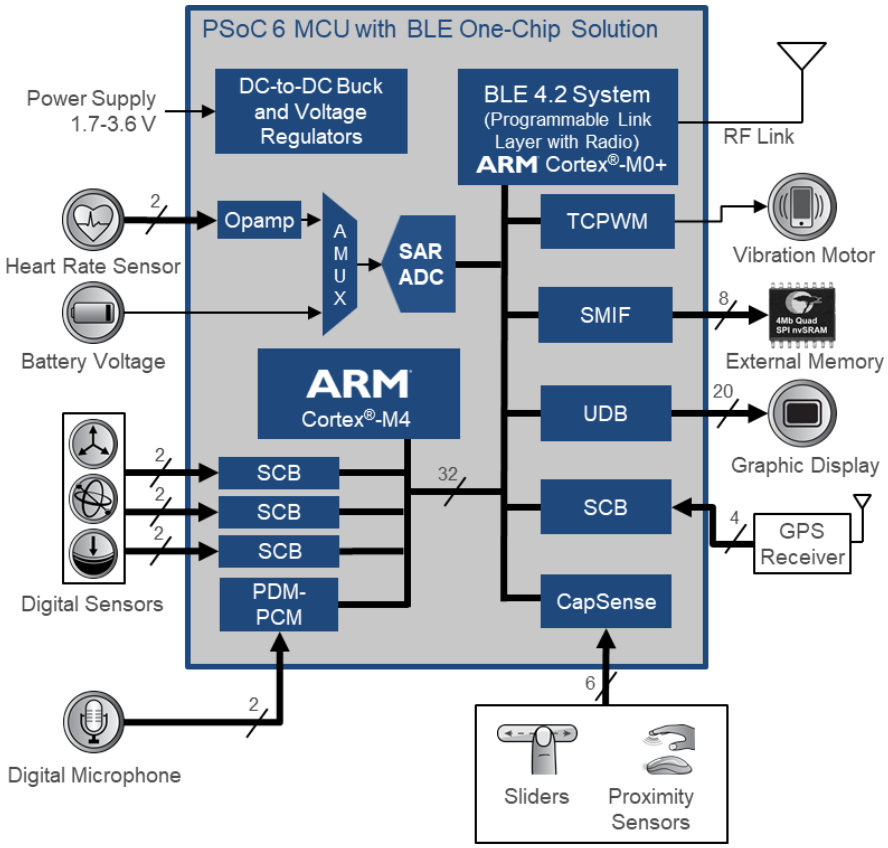
赛普拉斯在 PSoC 6 BLE 器件中的第四代电容式触摸感应功能（称为 CapSense）提供了前所未有的信噪比（SNR）;一流的防水功能;以及各种传感器类型，如按钮，滑块，跟踪板和接近传感器。 CapSense 用户界面在可穿戴电子设备（例如活动监视器和健康和健身设备）中越来越受欢迎。 CapSense 解决方案适用于嘈杂的环境和液体环境。

PSoC 6 BLE 可通过集成解决方案实现超低功耗连接应用。Figure 1 显示了基于 PSoC 6 BLE 的健身追踪器的应用级框图。该器件提供单芯片解决方案，包括以下内容：

- 低功耗 BLE 4.2 无线电，可同时支持四个连接
- 用于超低功耗操作的降压转换器
- 设备内的模拟前端（AFE），用于调节和测量心率传感器输出并监视电池电压
- 串行通信模块（SCB）与包括全球定位系统（GPS）模块的多个数字传感器接口

- □脉冲密度调制（PDM）脉冲编码调制（PCM）硬件引擎和用于语音的数字麦克风接口
- □采用 CapSense 技术实现可靠的触摸和接近感应
- □串行存储器接口（SMIF），支持通过四线串行外设接口（QSPI）启用的外部存储器
- □数字逻辑（UDB）和外设（TCPWM）驱动显示和振动电机

Figure 1. 健身追踪器应用框图



本应用笔记介绍了 PSoC 6 BLE 的功能，概述了开发生态系统，并讲解如何开始一个简单的设计：带有即时警报服务 Immediate Alert Service (IAS)的 Bluetooth SIG 标准 Find Me Profile (FMP)。 对于 Cypress 套件 CY8CKIT-062-BLE，此设计可用作代码示例 CE212736。 本应用笔记大量使用代码示例。

有关硬件设计注意事项，请参考 AN218241 – PSoC 6 MCU 硬件设计注意事项。

1.1 先决条件

开始进行前，请保证您已经具有开发套件并已经安装好了所需软件，包括代码示例。

1.1.1 硬件

- CY8CKIT-062-BLE PSoC 63 with BLE Connectivity Pioneer Kit
- 使用 Windows 7 或更高版本的 PC (如果使用 CySmart™ Host Emulation Tool 应用)
- 使用 Android 5（或更高版本）或 iOS 8（或更高版本）的手机(如果使用 CySmart iOS/Android 应用)

1.1.2 软件

- PSoC Creator 4.2 或更高版本）以及 PSoC Programmer 3.27 （或更高版本）
- CE212736 – 针对 CY8CKIT-062-BLE Pioneer 套件的 *BLE Find Me* 代码示例
- CySmart Host Emulation Tool 电脑应用程序或 CySmart iOS/Android app。

手机 扫描以下二维码 下载 CySmart 应用程序.

iOS

Android

2 带 BLE 连接 资源的 PSoC 6 MCU

赛普拉斯网站 www.cypress.com 提供了丰富的数据 帮助您选择合适的 PSoC 器件，并快速有效地将其集成到您的设计中。 如果您是 PSoC 的 新手 ，建议您阅读 附录 C: PSoC 6 BLE 器件特性。

以下为 PSoC 6 BLE 简要列表:

- **概述:** PSoC Portfolio, PSoC Roadmap
- **产品选择:** PSoC 1, PSoC 3, PSoC 4, PSoC 5LP, PSoC 模拟协处理器, 或带 BLE 连接 PSoC 6 MCU . 此外，PSoC Creator 包括一个相似的器件选择工具为 PSoC Creator 项目选择器件。
- **数据手册** 为各个器件系列提供电气规范。
- **应用笔记和代码示例** 涵盖从基础到高级的各种主题。许多应用笔记还包括代码示例。 PSoC Creator 提供额外的代码示例—参见 代码示例.
- **技术参考手册(TRMs)** 提供各个器件 系列的架构和寄存器的详细描述。
- **CapSense 设计指南:** 了解如何使用 PSoC 器件 设计电容式触摸传感应用.
- **开发工具**
  - CY8CKIT-062-BLE PSoC 63 with BLE Connectivity Pioneer Kit 是一款易于使用且价格便宜的 PSoC 6 BLE 开发平台。此套件包括兼容 Arduino™屏蔽板和 Digilent® Pmod™子板的连接器。
  - 用于 Windows, iOS, 和 Android 的 CySmart BLE 主机仿真工具易于使用， 使您能够测试及调试你的 BLE 外设应用。

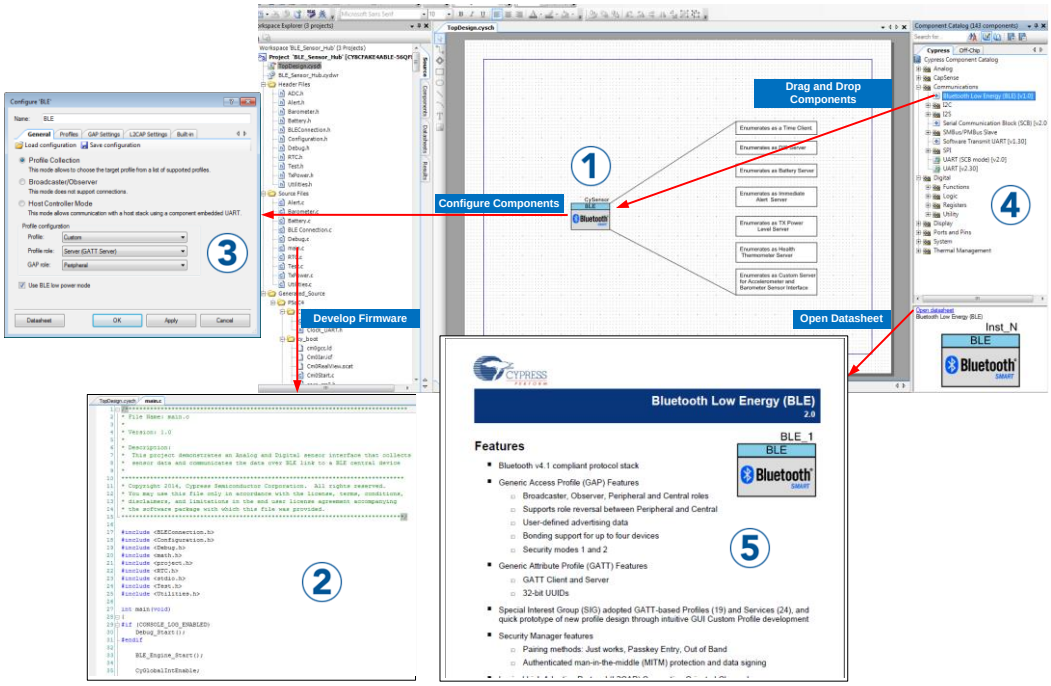
  参见赛普拉斯物联网开发工具了解工具概述。
- **培训视频:** 访问 www.cypress.com/training 获得 PSoC Creator 和 BLE 技术的各种资源。
- **技术支持**
  - Frequently Asked Questions (FAQs): 了解更多关于 BLE 生态系统的信息。
  - Forum: 查看你的问题是否在 PSoC 论坛得到其他开发者的回答。.
  - Cypress support: 访问我们的 support 页面并创建 技术支持案例 或联系 当地销售代表。 如果你在美国境内，你可以拨打我们的免费电话与技术支持团队沟通： +1-800-541-4736。按提示选择 8。

2.1 PSoC Creator

PSoC Creator 是一个免费的基于 Windows 的集成设计环境（ IDE 它使您能够基于 PSoC BLE 同时设计硬件和固件系统. 如 Figure 2 所示, 通过 PSoC Creator, 你可以：

1. 拖放组件以在主设计工作区中构建您的硬件系统设计.
2. 使用 PSoC 硬件同时设计你的应用固件。
3. 使用配置工具配置组件。
4. 浏览超过 100 个组件的组件库
5. 检查组件数据手册。

Figure 2. PSoC Creator 条目和组件



2.2 PSoC Creator 帮助

访问 PSoC Creator 主页下载并安装最新版本的 PSoC Creator。 然后 启动 PSoC Creator 并导航至以下项目：

- **快速入门指南**: 选择 **帮助** > **文档** > **快速入门指南** 。 本指南为您提供了开发 PSoC Creator 项目的基础知识 。
- **简单组件代码示例**: 选择 **文件** > **代码示例。** 这些代码示例展示了如何配置和实用 PSoC Creator 组件。
- **系统参考指南**: 选择**帮助** > **系统参考指南。** 该指南列出并描述了 PSoC Creator 提供的系统功能。
- **组件数据手册**: Right-click 右键点击组件并选择**打开数据手册。** 访问 PSoC 6 BLE 组件数据手册 页面，这里列出了所有组件。

2.3 代码示例

PSoC Creator 包含大量代码示例。 这些项目可从 PSoC Creator 的 **Start Page**（如 Figure 3 所示）或 **File> Code Example** ...菜单中获得。

代码示例可以通过完整的设计而不是空白页面加速您的设计过程。 代码示例还显示了 PSoC Creator 组件如何用于各种应用。 代码示例，数据表（文档选项卡）和示例代码都包含在内，如 Figure 4 所示。

在 Figure 4 所示的 **Find Code Example** 对话框中，你有如下几个选项：

- 根据器件系列过滤示例（本例中为 PSoC 4xxx BLE 或 PSoC 6 BLE）或关键字。
- 从 **Filter by** 选项的示例菜单中选择。 有几个 BLE 代码示例供您参考，如 Figure 4 所示。有些代码取决于平台，可能不会基于 **Device family** 选择出现。
- 查看选择的数据表（在“**Documentation**”选项卡上）
- 查看选择的代码示例。 您可以将此窗口中的代码复制并粘贴到您的项目中，这有助于加速代码开发。
- 根据选择创建一个新项目（如果需要，还可以是一个新的工作区）。 这可以通过完整的基本设计开始您的设计，从而加速您的设计过程。 然后，您可以将该设计应用于您的应用程序。
- 除了 PSoC Creator 代码示例外，您还可以在此 GitHub repository 中找到更多 BLE 参考示例。

Figure 3. PSoC Creator 中的代码示例

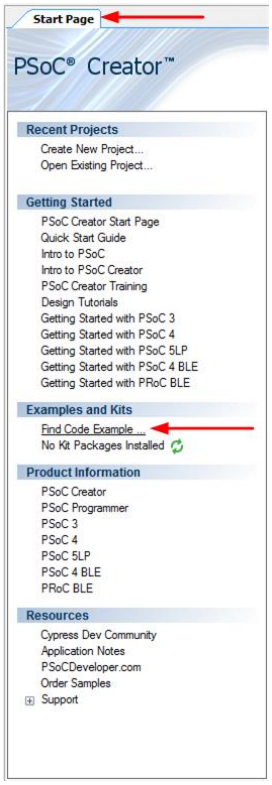
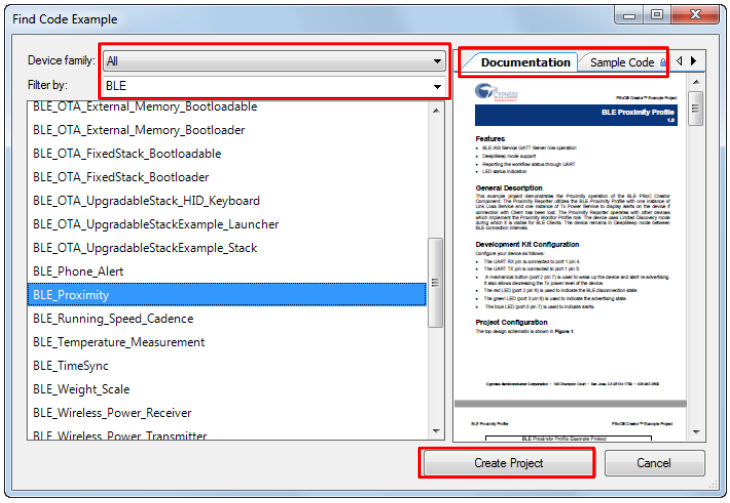


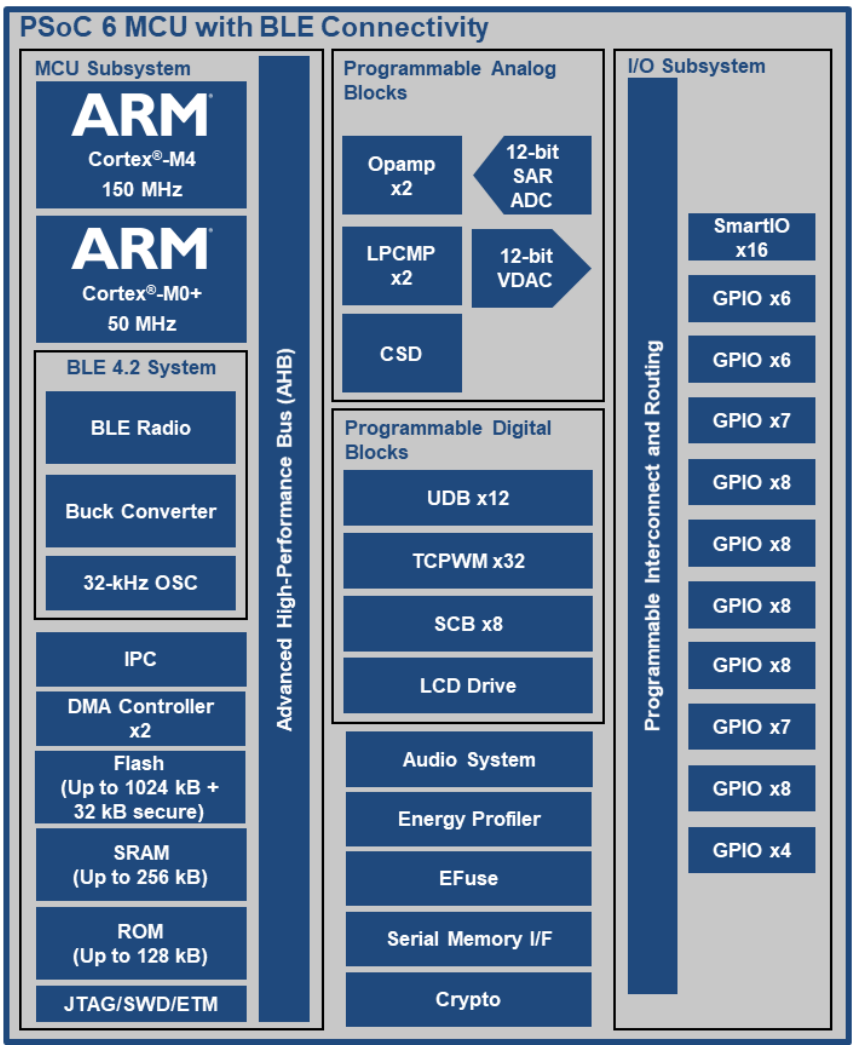
Figure 4.样本代码的代码示例



3 PSoC 6 MCU BLE 连接器件特性

PSoC 6 BLE 器件具有丰富的功能集，如 Figure 5 所示。以下列出了其主要功能。 有关更多信息，请参见器件数据手册，技术参考手册（TRM）以及相关应用笔记和代码示例部分。

Figure 5. PSoC 6 MCU BLE 连接框图



- **32 位双核 MCU 子系统**
  - 具有单精度浮点单元的 150MHz Arm Cortex-M4
  - 100-MHz Arm Cortex-M0+
  - 高达 1 MB 的闪存（32 KB 的 EEPROM 和 32 KB 的安全闪存）
  - 高达 288 KB 的 SRAM，在 32 KB 保留边界处具有可选的深度睡眠保留粒度
  - 硬件支持处理器间通信
  - 密码加速器，支持硬件加速和真随机数生成器功能
  - 两个 32 通道直接存储器访问（DMA）控制器
  - eFUSE: 用于安全密钥存储的一次性可编程位
  - 采用基于硬件散列的身份验证的不间断安全引导
  - 实时时钟

- **I/O 子系统**
  - 多达 78 个 GPIO,可用于模拟，数字，CapSense 或段 LCD 功能
  - 可编程的驱动模式，驱动强度和转换速率
  - 两个带智能 I / O 的端口，可以实现布尔操作
- **可编程模拟模块**
  - 两个 6 MHz 增益带宽（GBW）运算放大器，可配置为可编程增益放大器（PGA），比较器或滤波器
  - 两个低功耗比较器，其电流消耗低于 300 nA，可用于深度睡眠和休眠模式
  - 一个 12 位，1-Msps SAR ADC，带 16 通道时序器
  - 一个 12 位电压模式 DAC
- **具有 SmartSense™自动调谐功能的 CapSense**
  - 一个 CapSense Sigma-Delta（CSD）控制器，提供业内最佳的 SNR、液体容差和接近感应
  - 一个互电容感应 CapSense 发送/接收（CSX）控制器
- **可编程数字模块，通信接口**
  - 用于定制数字外设的 12 个通用数字模块（UDB）
  - 32 定时器/计数器/脉宽调制器（TCPWM）模块可配置为 16 位/ 32 位定时器，计数器，PWM 或正交解码器
  - 9 个 SCB 可配置为 I^2C 主机或从机，SPI 主机或从机或 UART
  - 具有一个 I^2S 接口和两个 PDM 通道的音频子系统
  - SMIF 接口，支持来自外部四通道 SPI 闪存的现场执行和即时加密和解密
- **使用蓝牙低功耗 4.2 的蓝牙连接**
  - 带集成巴伦的 2.4GHz BLE 无线信号
  - 符合蓝牙 4.2 规范的控制器和主机实施
  - 链路层引擎，支持主/从模式和最多四个同时连接
  - 支持 2 Mbps LE 数据速率
- **工作电压范围，电源域和低功耗模式**
  - 器件工作电压：1.71 V 至 3.6 V
  - 用户可选择的 1.1 V 或 0.9 V 的核心逻辑操作
  - 多个片上稳压器：低压差输出（用于有源、深度睡眠模式的 LDO），单输入多输出（SIMO）降压转换器
  - 活动，低功耗活动，睡眠，低功耗睡眠，深度睡眠和休眠模式，可实现精细电源管理
  - 具有可操作 BLE 链路的深度休眠模式：4.5 V 典型电流，3.3 V，64 KB SRAM 保留
  - 具有内置实时时钟（RTC）、电源管理集成电路（PMIC）控制和有限 SRAM 备份的“永远在线”备用电源域

4 具备 BLE 连接开发生态系统的 PSoC 6 MCU

4.1 配置 BLE 连接先锋套件的 PSoC 63

带有 BLE 连接先锋套件的 PSoC 63 是赛普拉斯的 BLE 开发套件，支持 PSoC 63BL 系列器件。 有关更多信息，请参阅赛普拉斯物联网开发工具中的带 BLE Pioneer 先锋套件的 PSoC 63。

4.2 固件/应用开发

赛普拉斯 PSoC Creator 和外设驱动库（PDL）是开发过程的核心。

PDL 是 PSoC 6 MCU 系列的软件开发套件。作为源代码提供的 PDL 使得为支持的设备开发固件变得更加容易。它可以帮助您快速定制和构建固件，而不需要了解寄存器组。 PDL 支持 PSoC Creator 和第三方 IDE。

PSoC Creator 是一款开发工具，可用于协同设计设备硬件和固件。 PSoC Creator 中有超过 100 个预编译和配置的组件。这些组件中的许多都基于 Peripheral Driver Library，因此可以轻松配置和定制外设驱动程序。

PSoC 6 BLE 系列器件的应用开发遵循类似于其他 PSoC 系列的方法。 PSoC Creator 汇集了多个数字/模拟/系统组件和固件来构建应用程序。使用 PSoC Creator，您可以选择、放置和配置原理图上的元件;编写 C /汇编源代码;并编程和调试设备。

借助 PSoC 6 BLE 器件，PSoC Creator 全面支持导出器件配置，以便您可以在熟悉的 IDE 中继续进行固件开发。您可以在 PSoC Creator 中添加 TopDesign 原理图，并在器件中配置硬件。选择您的目标 IDE 并生成相应的源代码。然后将生成的源代码包导入到 IDE 中。请参阅支持其它 IDE。

针对 PSoC 6 BLE 开发的固件代码可以针对 Cortex-M4 内核或 Cortex-M0 +内核运行。此定位可以在 PSoC Creator 工作区浏览器的源文件级完成。

4.3 外设驱动库

PDL 是用于 PSoC 6 BLE 器件的软件开发套件。

希望在寄存器级别工作的固件开发人员也应该安装 PDL。PDL 包含您的项目所需的所有设备特定的头文件和启动代码。它也可以作为每个驱动的参考。由于 PDL 是作为源代码提供的，因此您可以看到它如何访问寄存器级别的硬件。

有些设备不支持特定的外设。 PDL 是为任何受支持设备构建驱动程序所需的所有代码的超集。这个超集设计意味着：

- 初始化、配置和使用外设所需的所有 API 都可用。
- 无论可用的外设如何，PDL 都可用于所有 PSoC 6 MCU 器件。
- PDL 包含错误检查功能，以确保目标外设存在于所选设备上。

  这样，只要外设可用，代码就可以保持跨平台的兼容性。设备头文件指定可用于设备的外设。如果您编写的代码试图使用不支持的外设，则编译时会出现错误。在编写代码以使用外设之前，请查阅特定器件的数据表以确认对外设的支持。

  PSoC Creator 为您提供了基于 PDL 的组件，可用于 PSoC 6 BLE 器件。这保留了 PSoC Creator 在利用赛普拉斯或社区开发和预验证组件方面的精髓。但是，PDL 是一个源代码库，可以在任何开发环境中使用。

  PDL 包含以下关键软件资源：
- 每个外设驱动程序的头文件和源文件
- 中间件库的头文件和源文件，例如 BLE 和 CapSense
- 设备特定的头文件，启动和配置文件
- 支持的第三方 IDE 的模板项目
- 完整的文档

  文档的位置是 *<PDL install directory>\doc*\. 有两个关键文件。

  PDL v3.0 用户手册 涵盖了使用 PDL 的基础，例如：

  - 使用 PDL 创建自定义项目（包括第三方 IDE）
  - 配置外设

- 管理固件中的引脚
- 使用 PDL 作为基于寄存器编程的学习工具
- 使用 PDL API 参考文档

  第二个关键文档是 PDL 3.0 API Reference Manual.html。此参考手册包含关于 PDL 中每个驱动程序的完整信息，包括概述，配置注意事项以及每个函数，宏，数据结构和枚举类型的详细信息。

4.4 PSoC Creator

PSoC Creator 是一款免费的基于 Windows 的 IDE，支持：

- 完整的原理图截图的内部硬件设计
- 经过预先验证的生产就绪组件，可访问设备资源，例如：
  - 系统：时钟，中断，GPIO，DMA
  - 通信：BLE, I^2C, LIN, SPI, USB, UART
- 在 Verilog 中开发自定义组件和/或使用状态机图表的工具
  - PDL
- 集成的 C 源代码编辑器和编译器
- 通用编译器，Arm MDK, Arm DS
- 内置调试器

该工具可帮助您配置和编程 PSoC 器件中的外设。 PDL 外设可能有很多选项。 PSoC Creator 界面使配置和定制驱动程序变得更加简单。 PSoC Creator 无缝地生成所有必需的 PDL 配置结构。

有关如何使用 PSoC Creator 进行系统开发的信息，请参见 PSoC Creator User Guide.

4.5 支持其它 IDE

您还可以使用您最喜爱的 IDE（例如 IAR Embedded Workbench）为 PSoC 6 BLE 开发固件。在较高层次上，您可以通过以下两种方式之一将 PDL 与另一个 IDE 一起使用：

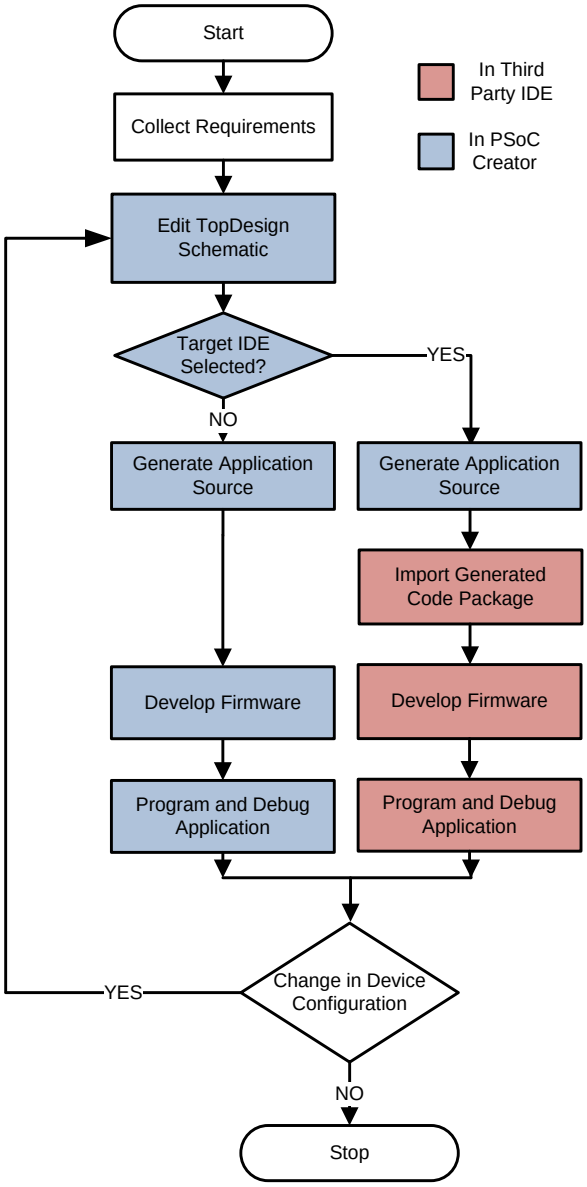
- 使用 PSoC Creator 设计系统并生成配置代码;然后导出到目标 IDE
- 完全在 IDE 中工作，根本不使用 PSoC Creator

有关详细信息，请参见 AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project 。

4.5.1 使用 PSoC Creator 来对标另一个 IDE

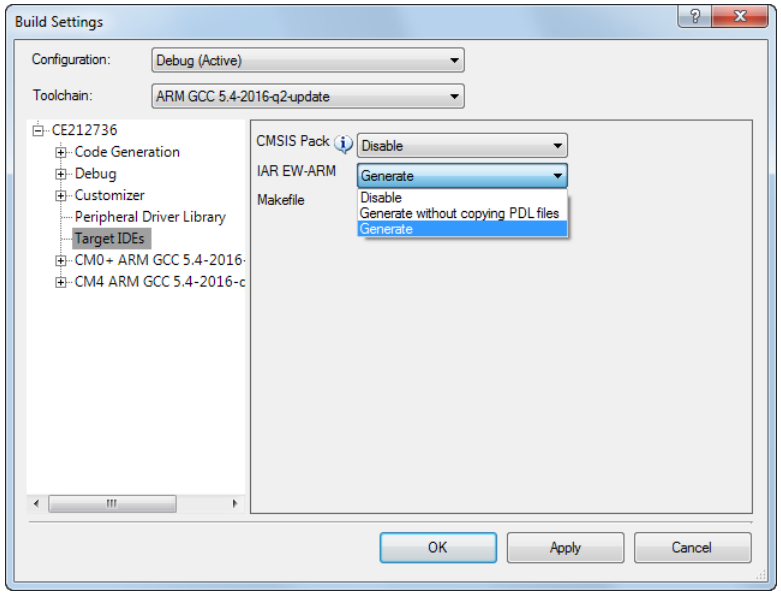
赛普拉斯建议您使用 PSoC Creator 来设置和配置 PSoC 6 BLE 系统资源和外设。然后将项目导出到 IDE，并继续在 IDE 中开发固件。如果设备配置发生变化，请在 PSoC Creator 中编辑 TopDesign 原理图并重新生成目标 IDE 的代码。Figure 6 显示了工作流程。

Figure 6.使用 PSoC Creator 的 PSoC 6 BLE 应用开发流程图



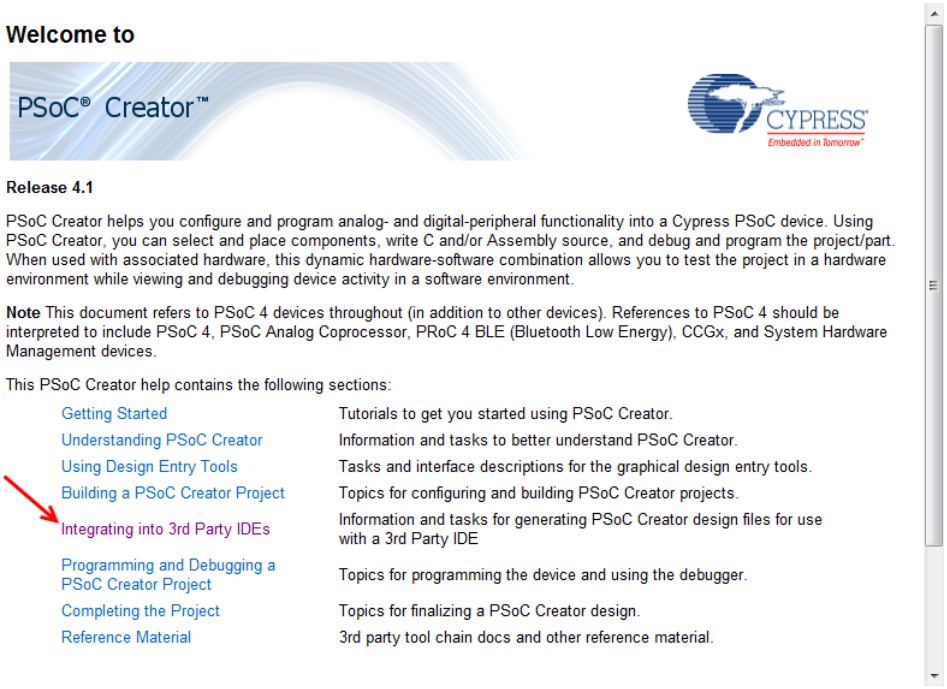
PSoC Creator 生成的代码包含基于您的设计的所有必需的头文件、启动和 PDL 文件。 Keil µVision，IAR Embedded Workbench，基于 Eclipse 的 IDE 以及基于 GNU 的命令行工具支持从 PSoC Creator 导出生成的代码。 您可以使用 Project> Build Settings> Target IDEs 面板选择目标 IDE。 为目标 IDE 启用导出包，如 Figure 7 所示。生成代码时，PSoC Creator 也会创建相应的导出包。

Figure 7. 在 Project > Build Settings 中选择目标 IDE



然后将包导入到 IDE 中。 每个 IDE 的导入细节差别很大。 请参阅 PSoC Creator 帮助以了解您必须遵循的过程。 选择帮助> PSoC Creator 帮助主题。 请参见 Figure 8. 集成到第三方 IDE 主题中指出了每个 PSoC 器件系列的特定帮助文件以及每个支持的 IDE。

Figure 8. PSoC Creator 帮助



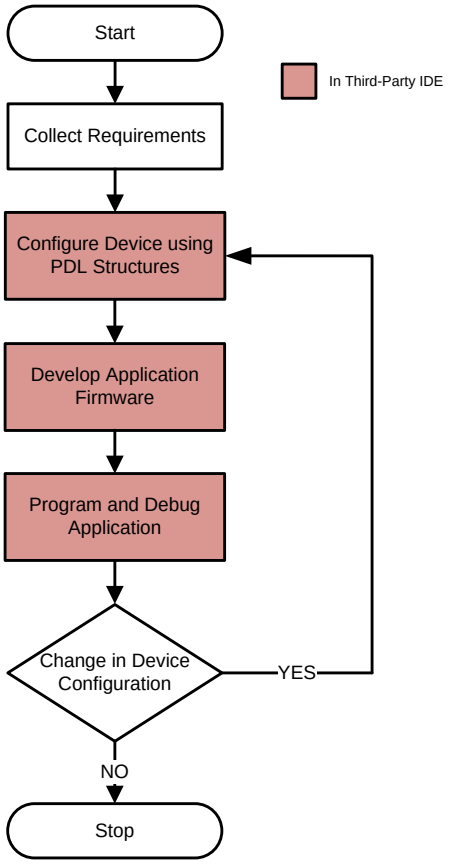
您可以在大多数（如果不是全部）IDE 中有效地工作。 如果您的 IDE 在 Target IDEs 面板中不受支持，仍然可以使用 PSoC Creator。 生成代码后，将必要的文件直接添加到 IDE 的项目中。

4.5.2 完全在第三方 IDE 中工作

您也可以使用第三方 IDE 来完成应用程序开发，而根本不使用 PSoC Creator。 在这种情况下，使用 PDL 作为源代码库，并将所需的任何文件添加到项目中。 但是，您无法使用 PSoC 6 器件上提供的任何 UDB。

在高层次上，在您的代码中，您可以使用 PDL 结构来配置设计中的外设。 在源代码中创建结构并初始化模块之后，您将像平常一样开发应用程序固件。 有关详细信息，请参阅 PDL v3.0 用户指南。 Figure 9 显示了涉及的过程的流程图。要更改设备的配置，请编辑配置结构。

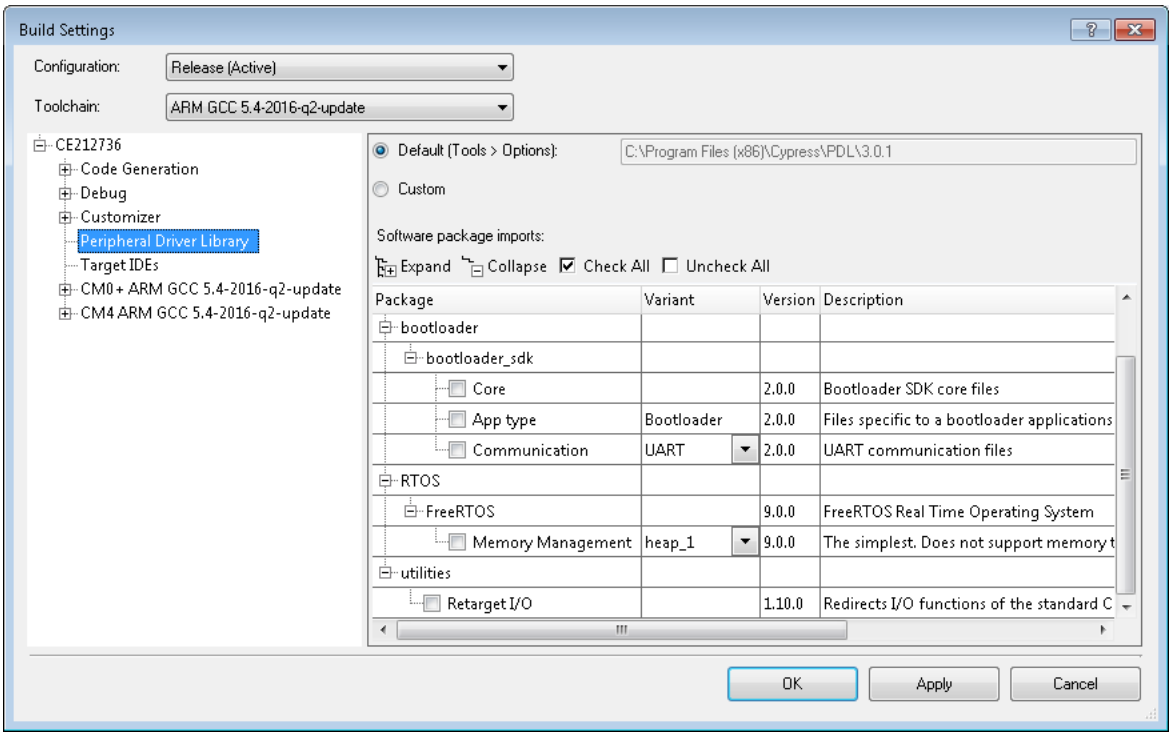
Figure 9.使用第三方 IDE 的 PSoC 6 BLE 应用开发流程图



4.6 RTOS 支持

PDL 包括用于 PSoC 6 BLE 开发的 RTOS 支持：RTOS 代码完全集成并包含在 PDL 中。 例如，FreeRTOS 的源文件可在<PDL 安装目录> \ rtos \文件夹中找到。 您还可以使用 PSoC Creator RTOS 导入选项将 FreeRTOS 软件包导入到项目中。 导航到 **Project> Build Settings** 菜单，并从 **Software package imports：**选项中选择 **RTOS> FreeRTOS**。

Figure 10. 在 PSoC Creator 项目中导入 FreeRTOS



如果您有自己的首选 RTOS，请使用提供的资源作为示例，介绍如何将此类代码与 PDL 集成。

4.7 调试

调试支持因工具而异。带 BLE Pioneer 先锋套件的 PSoC 63 具有 KitProg2 板载编程器/调试器。 它支持 CMSIS-DAP 和自定义操作模式，以及 KitProg2 连接。 这使得对 PSoC 6 BLE Pioneer 套件的调试非常灵活。 有关详细信息，请参阅 KitProg2 User Guide。

PSoC Creator 支持一次调试单个内核（Cortex-M4 或 Cortex-M0 +）。 一些第三方 IDE 支持多核心调试。 有关使用 PSoC Creator 调试 PSoC 器件的更多信息，请参考 PSoC Creator 帮助。

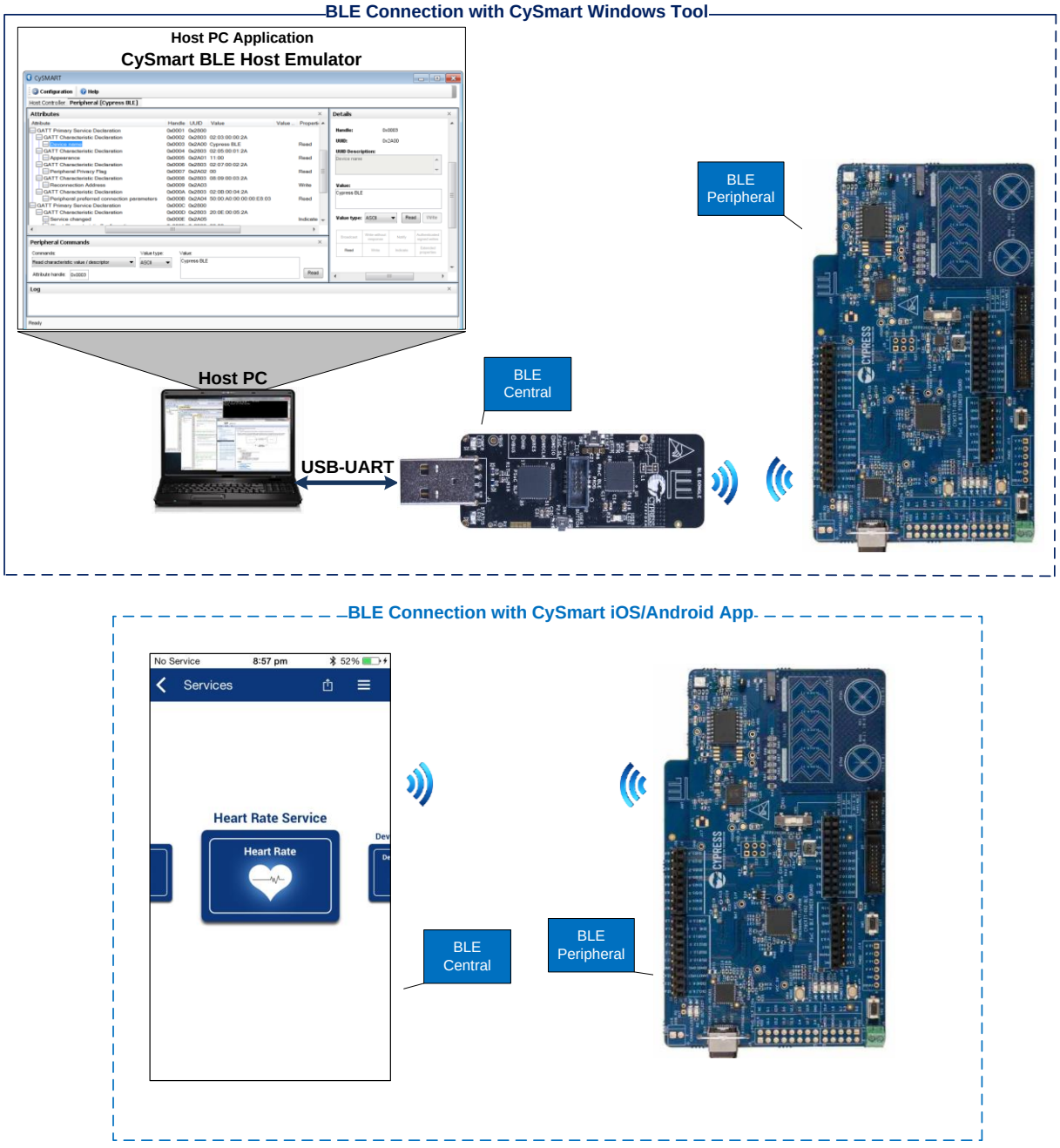
4.8 CySmart 主机仿真工具和移动应用程序

CySmart 主机仿真工具是一个 Windows 应用程序，它使用 PSoC 6 BLE Pioneer 套件的 BLE 软件狗模拟 BLE 中央器件。 有关更多信息，请参阅附录 D 中的 CySmart。 用于 iOS 和 Android 设备的 CySmart 具有类似的功能。 该工具在测试 BLE 应用程序时非常有用。

5 PSoC 6 MCU BLE 连接性开发设置

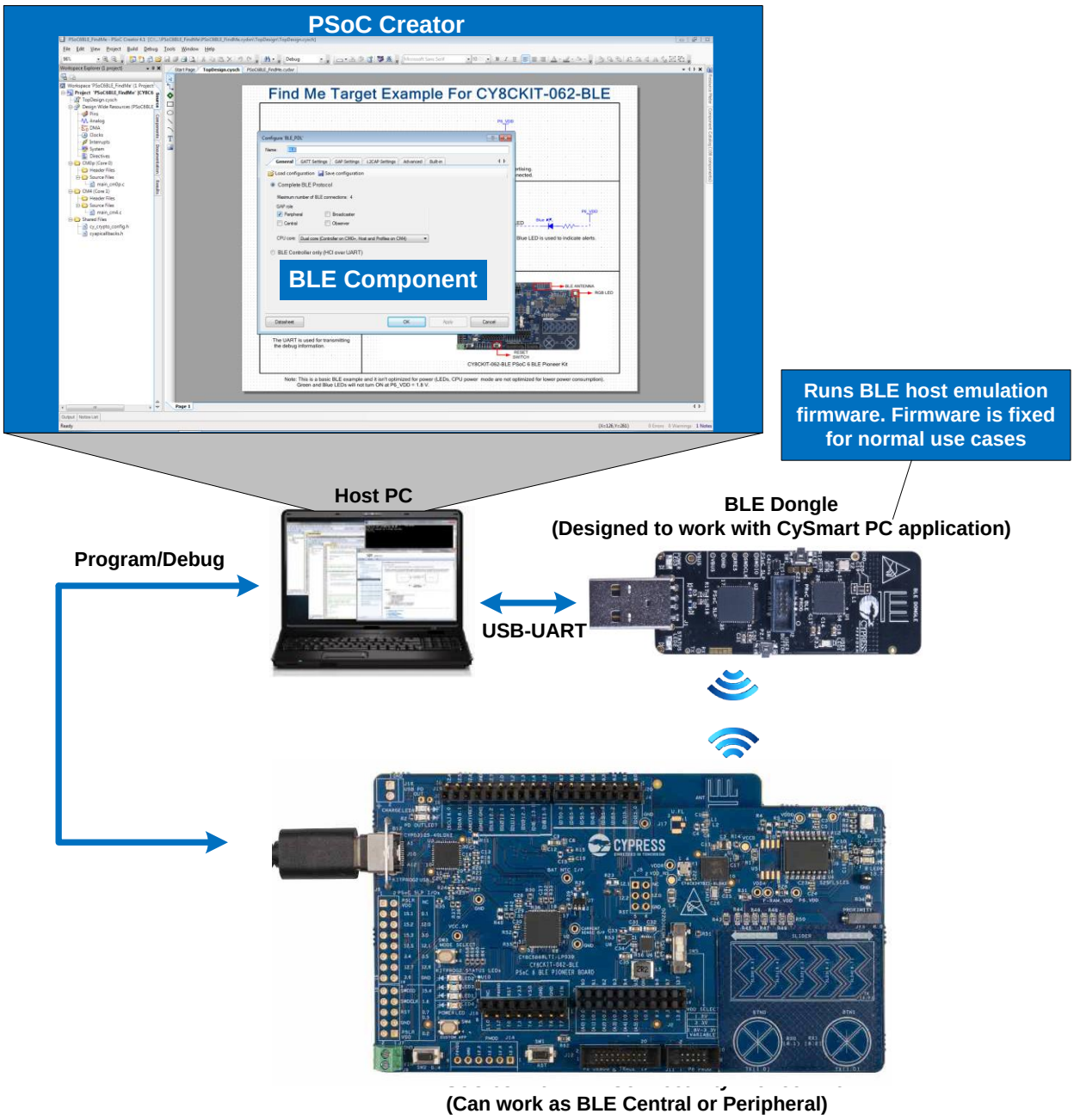
Figure 11 显示了使用 PSoC 6 BLE 器件评估 BLE 外设设计所需的硬件和软件工具。 在典型的使用案例中，PSoC 6 BLE Pioneer 套件（Figure 11 中的蓝板）被配置为可与 CySmart iOS / Android 应用程序或 CySmart 主机仿真工具等中央器件进行通信的外设。 CySmart 主机仿真工具还需要一个 BLE 软件狗（Figure 11 中的黑板）才能操作。 PSoC 6 BLE Pioneer 套件包含一个加密狗。

Figure 11. BLE 功能设置



如 Figure 12 所示，BLE 软件狗被预编程为与 Windows CySmart 主机仿真工具一起工作。 PSoC 6 BLE Pioneer 套件具有板载 USB 编程器，可与 PSoC Creator 配合使用，用于编程或调试 BLE 设计。 BLE Pioneer 套件可通过 USB 接口供电。 BLE 加密狗和 BLEPioneer 套件都可以同时连接到通用主机 PC 进行开发和测试。

Figure 12. BLE 开发设置



6 我的第一款采用 BLE 的 PSoC 6 MCU 设计

本节提供逐步说明，为 PSoC 6 BLE Pioneer 套件构建简单设计。 设计中实现了简单的 Bluetooth SIG 定义的标准配置文件，称为 Find Me Profile (FMP)。

虽然 PSoC 6 BLE Pioneer 套件旨在简化和简化基于 BLE 的应用程序的设计，但您也可以使用此设置来开发不使用 BLE 的应用程序。

6.1 使用以下指令

这些说明分为几个部分。 每个部分都致力于应用程序开发工作流程的特定阶段。 主要部分是：

- Part 1:
- Part 2:
- Part 3:
- Part 4:
- Part 5:
- Part 6:

这些说明要求您使用特定的代码示例。 但是，使用代码示例的程度取决于您选择遵循这些说明的路径。

我们通过这些说明定义了三条路径：

路径	从零开始工作 代码示例仅供参考	PSoC Creator 或 BLE 新手使用代码示例	熟悉 PSoC Creator 和 BLE 时使用代码示例
最适合	希望亲自体验 PSoC Creator 和/或 BLE 的人	对这个工具或技术陌生的人，并想看看它是如何工作的	知道这些工具和技术，并希望看到它在 PSoC 6 平台上工作的人

在每个部分的开始部分都明确定义了每条路径需要做的事情。

如果您从头开始并遵循本应用笔记中的所有说明，则可以按照说明使用代码示例作为参考。 从零开始工作会告诉您关于 PSoC Creator 设计过程以及 BLE 如何工作的最多信息。 这条路径也需要最多的时间。

您也可以直接使用代码示例。 这是一个完整的设计，所有的固件都已经写好。 您可以阅读说明并观察代码示例中如何实施这些步骤。如果您已经熟悉 PSoC Creator，并且想要了解如何配置 BLE 组件，这是方法特别有用。 如果直接使用代码示例，则可以跳过一些步骤。

在所有情况下，您都应该先阅读准备工作和关于设计。

6.2 准备工作

确保您的主机上安装了以下项目。

- PSoC Creator 4.2 或更新版本
- PDL v3.0.1 或更新版本
- CySmart Host Emulation Tool 或 CySmart iOS/Android 应用程序
- CE217236 – 此应用笔记配套的代码示例
- CY8CKIT-062-BLE PSoC 63 with BLE Connectivity Pioneer Kit，这个套件上执行了示例代码

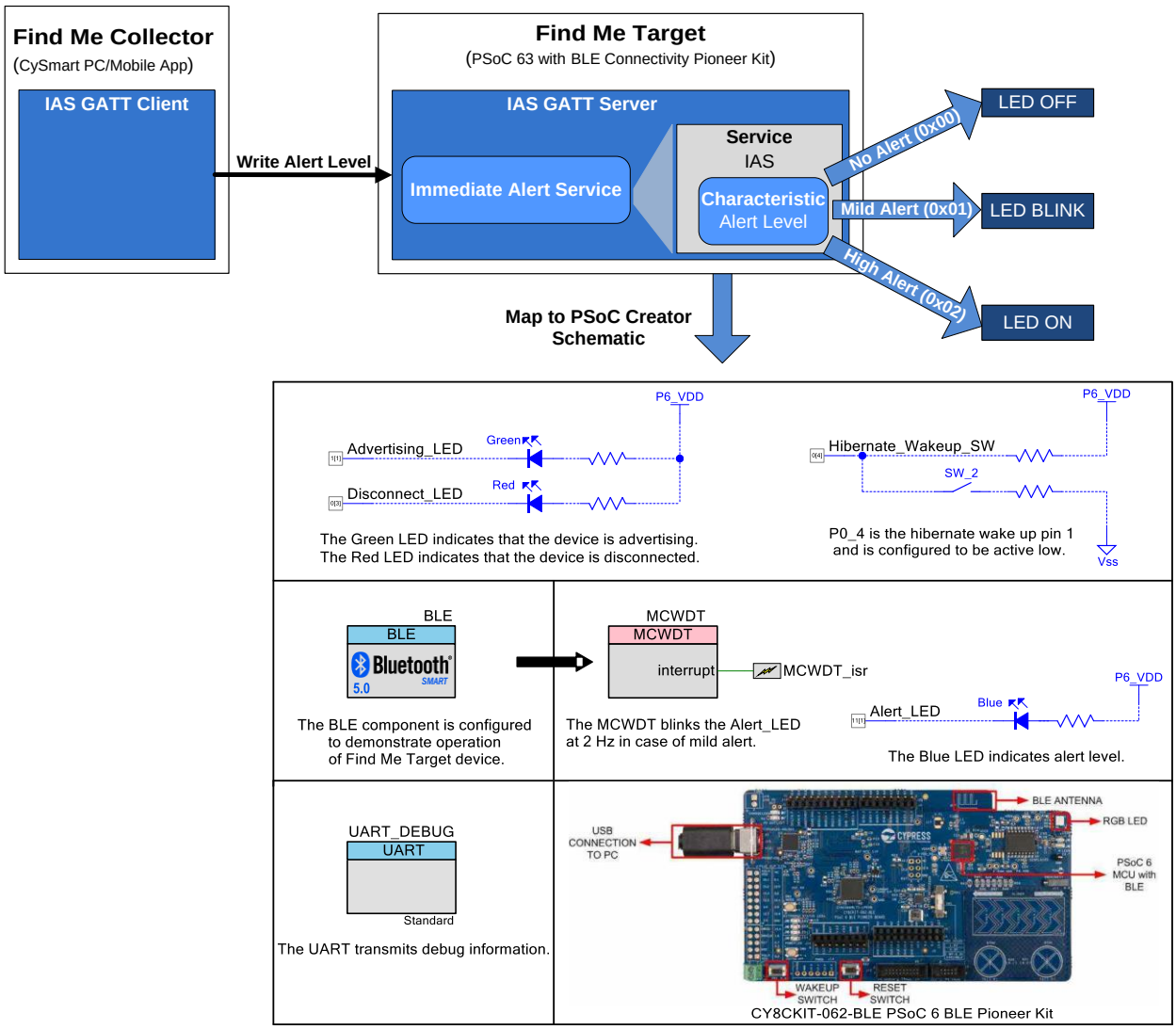
您可以点击上面的链接从赛普拉斯网站下载代码示例。 您还可以使用 **File** > **Code Example** 命令。 将器件系列设置为 PSoC 6 BLE。 然后将“**Filter by**”选项设置为“**Find Me**”。 出现 BLE_FindMe 代码示例。 选择代码示例并单击 **Create Project**，然后按照屏幕上的说明进行操作。

此设计是为 CY8CKIT-062-BLE PSoC 63 with BLE Connectivity Pioneer Kit 而开发的。 如果您希望使用其他硬件，则必须根据您的需要调整说明。

6.3 关于设计

此设计在由立即警报服务（IAS）组成的目标角色中实施 BLE Find Me Profile (FMP)。 FMP 和 IAS 是由 Bluetooth SIG 定义的 BLE 标准配置文件和服务。 Find Me 定位器触发的警报级别通过改变 BLE Pioneer 套件上 LED 的状态来指示，如 Figure 13 所示。 另外，两个状态 LED 指示 BLE 接口的状态。 还有一个可选的 UART 组件将调试消息打印到终端窗口。

Figure 13. 我的首个 PSoC 6 BLE 设计



6.4 Part 1:从零开始创建新项目

本部分将逐步介绍从零开始创建项目文件的过程。

路径	从零开始工作 代码示例仅供参考	PSoC Creator 或 BLE 新手使用代码示例	熟悉 PSoC Creator 和 BLE 时使用代码示例
操作	执行所有步骤	执行 Step 1 阅读其它步骤	执行步骤 1，然后跳转到 Part 2

注意：这些说明假定您使用的是 PSoC Creator 4.2 或更高版本。 PSoC Creator 的后续版本的整体开发过程相同; 但用户界面可能随时间而改变。

启动 PSoC Creator 并开始操作。

1. **确保 PSoC Creator 能够找到 PDL.**

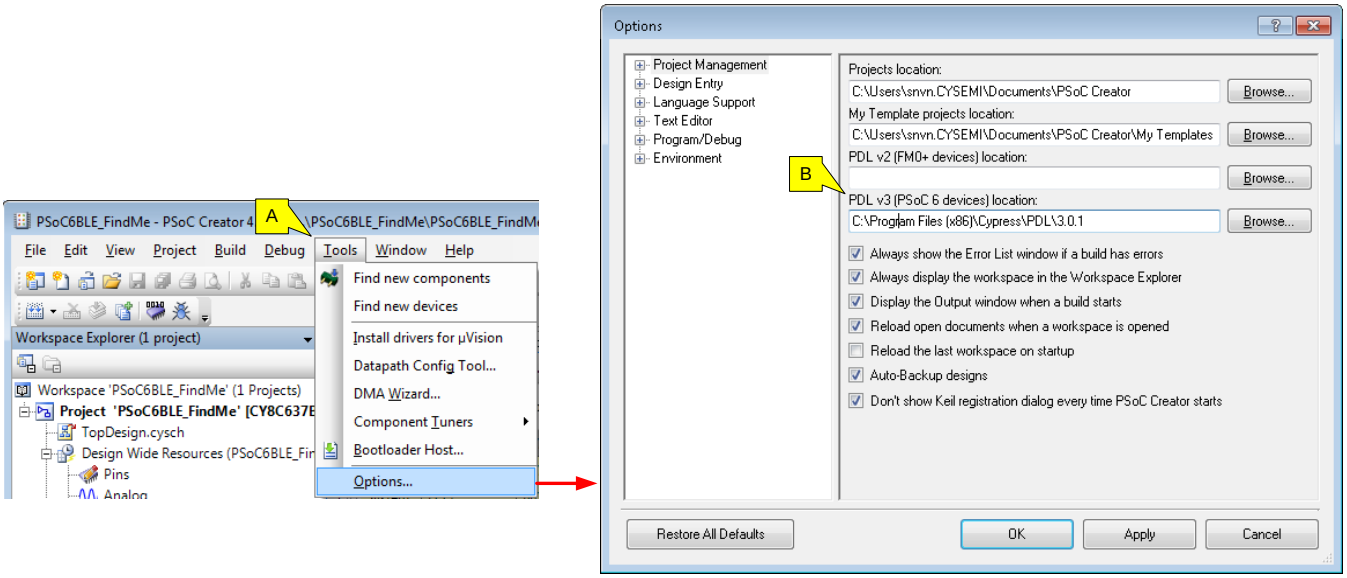
   这应该在安装过程中自动正确设置，但如果设置不正确，则无法正常工作。 有关此步骤的帮助，请参阅 Figure 14。

   A. 选择菜单项 **Tools** > **Options**

   B. 在“**Project Management**”面板上，检查 PDL v3（PSoC 6 器件）位置字段中的路径。

   C. .确保路径是正确的。 如果不是，请单击“**Browse**”按钮并找到已安装的 PDL 目录。 默认位置是 *C:\Program Files (x86)\Cypress\PDL\3.0.1*.

Figure 14. 外设驱动库 (PDL) 位置

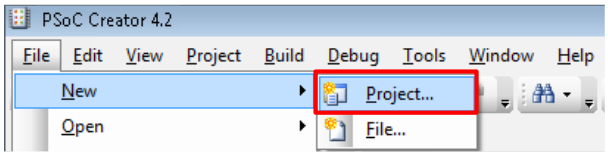


可选操作：：跳转到 Part 2: .

2. 创建新的 PSoC Creator 项目

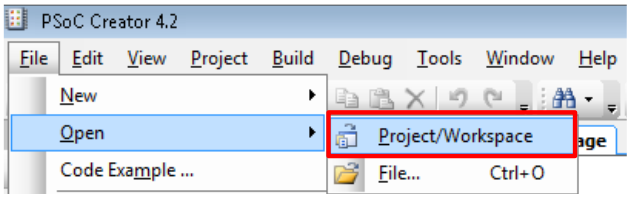
选择 **File** > **New** > **Project**, 如 Figure 15 所示. **Create Project** 窗口出现。

Figure 15. 创建新的 PSoC Creator 项目



注意：如果您使用的是代码示例，请选择 **File** > **Open** > **Project/Workspace**，如 Figure 16 所示。 **Open** 窗口出现。 指向代码示例工作区的位置并打开工作区。

Figure 16.打开现有代码示例工作区



3. 选择 PSoC 6 BLE 作为目标器件.

PSoC Creator 通过为指定的开发套件或目标器件自动设置各种项目选项，加快了开发进程。 有关此步骤的帮助，请参见 Figure 17。

A. 点击 **Target device**.

B. 在系列下拉菜单中，选择 **PSoC 6**.

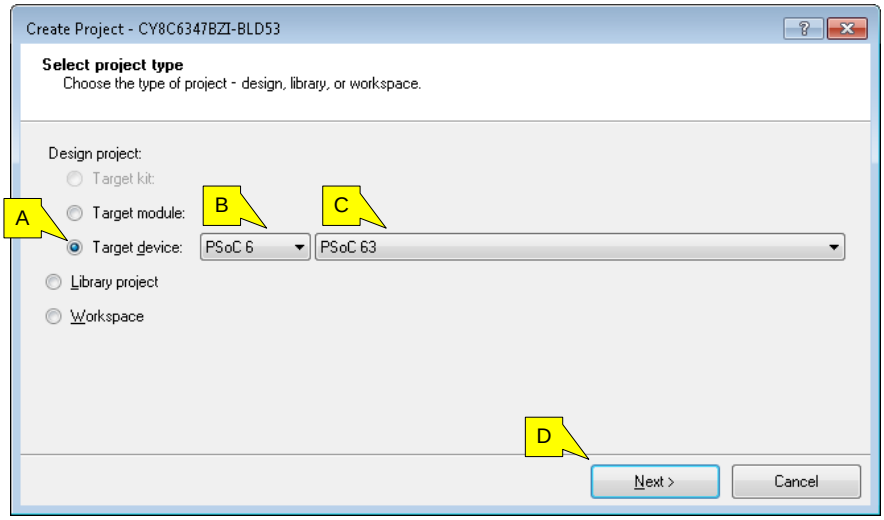
D. 在器件下拉菜单中，选择 **PSoC 63**.

E. 点击 **Next**. 选择项目模板的面板出现。

PSoC Creator 使用 CY8C6347BZI-BLD53 作为 BLE 系列 PSoC 6 MCU 中的默认器件。 该器件安装在有 BLE 连接先锋套件的 CY8CKIT-062-BLE PSoC 63 上。

如果您使用的是基于 PSoC 6 BLE 或不同 PSoC 6 BLE 部件号的自定义硬件，请选择 **Target device**（目标设备）中的 **Launch Device Selector**（启动设备选择器）选项，然后选择适当的部件号。

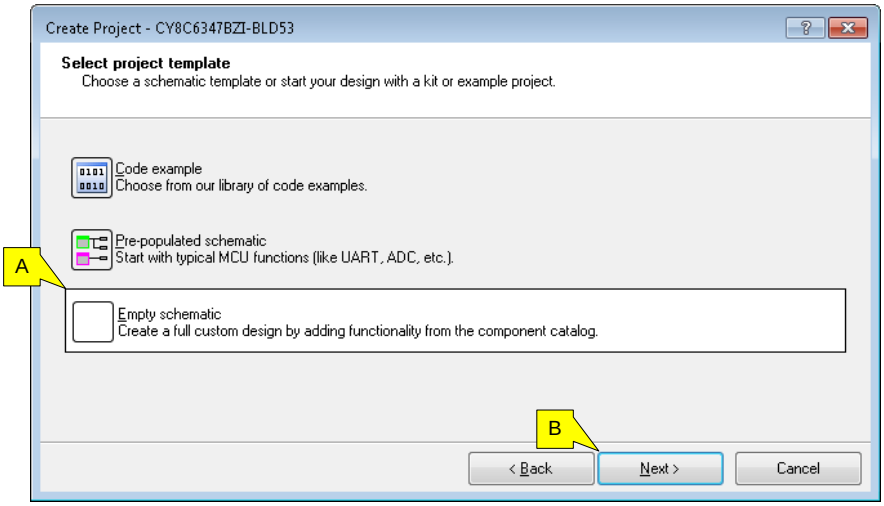
Figure 17. 选择目标器件



4. 选择项目模板

选择空白原理图并点击 **Next。**

Figure 18. 选择项目模板

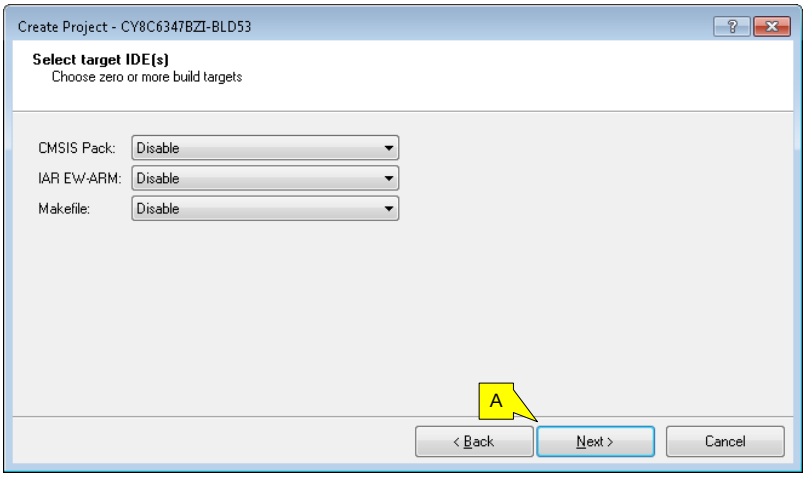


5. 选择目标 IDE

如果您希望从项目中导出代码，请指定目标 IDE。 默认情况下，所有导出选项都被禁用。 如果情况发生变化，您可以稍后修改此设置。

单击 **Next** 以接受默认选项。

Figure 19. 选择目标 IDE (全被禁用)



6. 创建项目

在此步骤中，为您的工作场所设置名称和位置，并为项目设定名称。 请参阅 Figure 20 以获取此步骤的帮助。 工作区是一个或多个项目的容器。

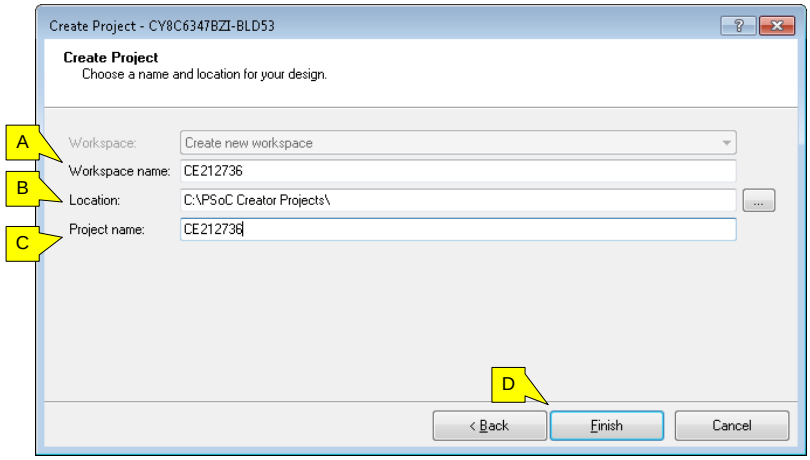
A. 设置工作区名称。

B. 指定工作区的位置。

C. 设置项目名称。 项目和工作区名称可以相同或不同。

D. 点击 **Finish**.

Figure 20. Project 名称和位置



您已经成功创建了一个新的 PSoC Creator 项目。

6.5 Part 2: 进行设计

既然您有一个项目文件，现在就可以使用 PSoC Creator 组件进行硬件设计了。 如果您直接使用代码示例，那么您已经拥有完整的设计。 通过本练习，根据您选择的路径执行建议的操作。

路径	从零开始工作 代码示例仅供参考	PSoC Creator 或 BLE 新手使用代码示例	熟悉 PSoC Creator 和 BLE 时使用代码示例
操作	执行所有步骤	阅读并理解所有步骤	如果愿意，你可以跳过这个部分。 跳转到 Part 3:

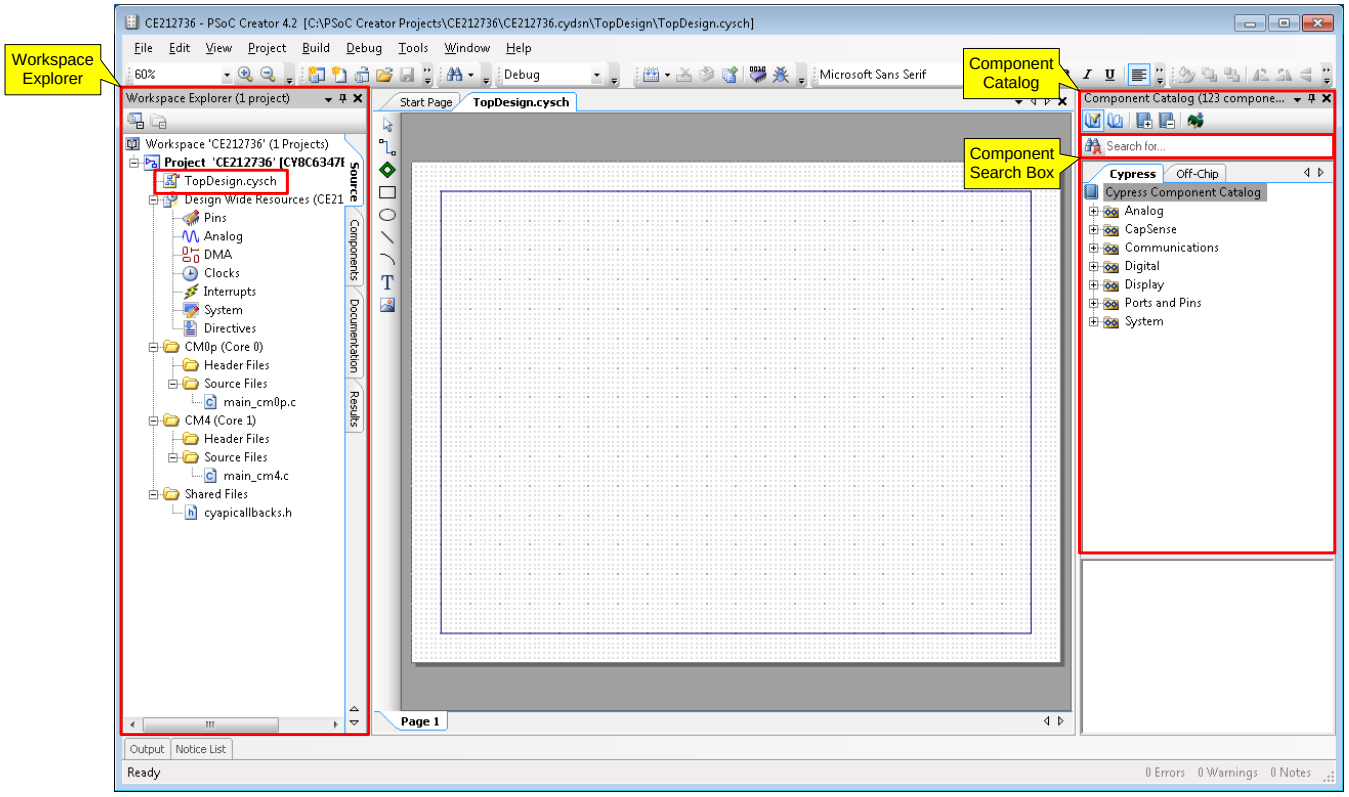
在实施设计之前，请按顺序快速浏览 PSoC Creator 界面。

Figure 21 为展示空白设计原理图的 PSoC Creator 应用程序。

该项目包含一个包含一组基本文件的项目文件夹。 您可以在左侧的“**Workspace Explorer**”窗格中查看这些文件。 项目原理图默认打开。 这是 TopDesign.cysch 文件。 双击资源管理器窗格中的文件名称以随时打开原理图。 在新项目中，原理图是空的。 如果您使用的是代码示例，这是 Find Me BLE 应用程序的原理图。

组件目录位于窗口的右侧。 您可以使用 **View> Component Catalog** 菜单项打开它。 您可以通过在 **Search for…**文本框中输入组件的名称，然后按下回车键来搜索特定的组件。 参见 Figure 21。

Figure 21. 原理图和组件目录



1. 在设计中放置组件

该设计使用多个组件：蓝牙低功耗组件，三个数字输出引脚，一个 UART，一个看门狗定时器和一个中断。 在这一步中，您将它们添加到设计中。 您可以在后续步骤中配置它们。 Figure 22 显示了结果。

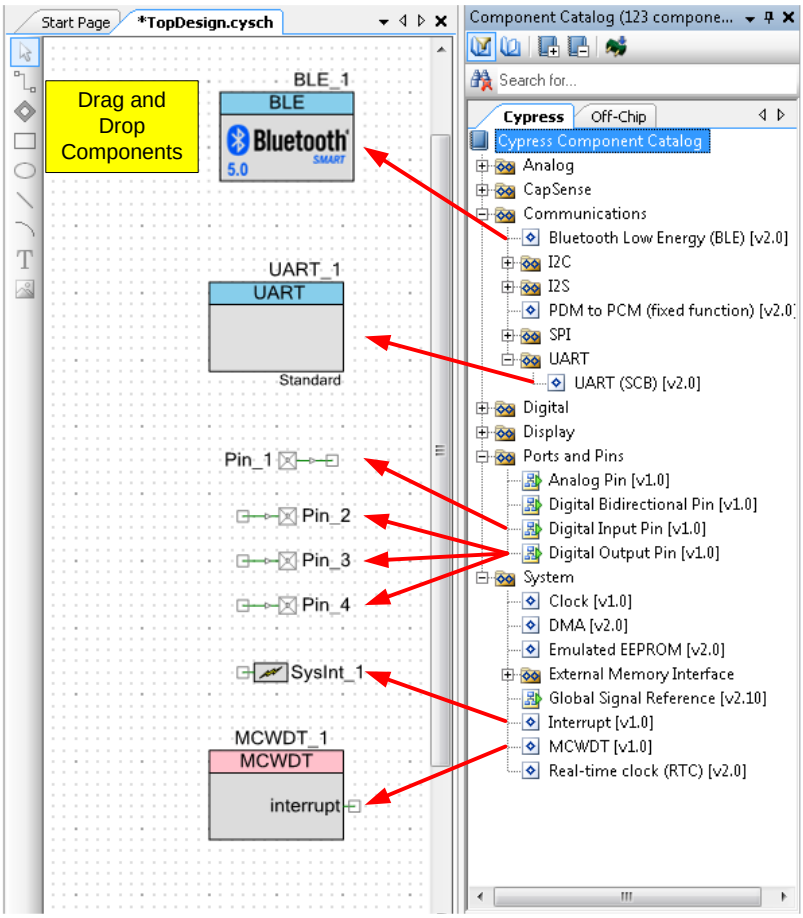
A. 在 **Component Catalog**（组件目录）中，展开 **Communications**（通讯）组，将 BLE 组件拖放到原理图中，然后放下。 放置组件的位置并不重要。 或者，您可以通过在组件目录搜索框中键入 BLE 来搜索 BLE 组件。

B. 同样在 **Communications** 组中，扩展 UART 并将 **UART（SCB）**组件拖放到设计中。

C. 展开 **Ports and Pins(**端口和引脚)组，将 **Digital Output Pin(**数字输出引脚)拖入设计中。 重复操作两次，实现总共三个引脚。

D. 展开 **System(**系统)组，将 **Interrupt(**中断)组件和 MCWDT 组件拖入设计中。

Figure 22.设计中放置组件



PSoC Creator 为每个组件提供默认名称和属性。 默认值可能适用于也可能不适用于任何给定的设计。 在后续步骤中，您可以修改名称和一些属性。

2. 配置三个 LED 引脚

一个引脚驱动警报状态 LED。 另外两个驱动 BLE 广告和断开指示器 LED。 BLE Pioneer 套件上的 LED 低电平有效; 也就是说，逻辑高引脚驱动状态会关闭 LED，逻辑低引脚驱动状态会将其打开。

除名称外，所有三个引脚都配置相同。 重复这些指令三次，每个针脚一次。 Figure 23 显示了配置。

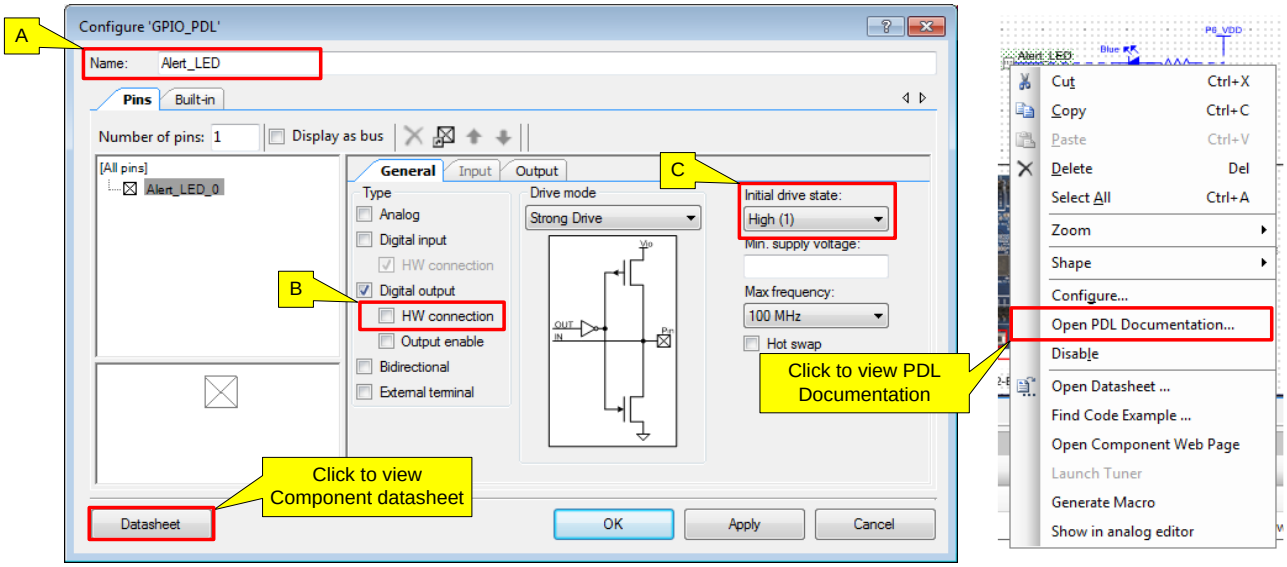
双击组件打开配置对话框。 然后执行以下步骤。

A. 更改每个引脚的组件实例的名称。 这三个名称分别是 Alert_LED，Advertising_LED 和 Disconnect_LED。

B. 对于每个引脚，取消选择硬件连接。 固件将驱动引脚。

C. 对于每个引脚，将 **Initial drive state**（初始驱动状态）设置为 **High (1)**。 因此，默认情况下，LED 将熄灭。

Figure 23.配置输出引脚组件



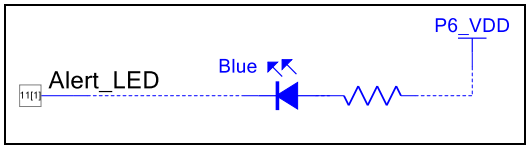
确保你配置了所有三个引脚。

提示：每个组件都有一个可从配置窗口访问的关联数据表。组件数据表提供了有关组件配置，应用程序编程接口（API）和电气规格的更多信息。

提示：通过右击组件并点击 **Open PDL Documentation...**链接，可以打开组件的相关 PDL 驱动程序的数据表。参见 Figure 23。

提示：对于引脚，如果启用 **External terminal(**外部终端**)**，则可以将外部“片外”组件添加到设计中。包含在原理图中的外部组件仅用于描述目的;它们对生成的代码没有影响。片外组件是可选的，但可以帮助硬件设计团队了解设计的工作原理。您还可以将文本框说明添加到设计中。Figure 24 显示了如何增强 Alert LED 的设计。在这种情况下，片外组件被配置为 Instance_Name_Visible 未选中。电阻器的 **Value** 字段保留为空白。 电源终端配置：Supply_Name 被设置为 P6_VDD。

Figure 24. 片外组件的输出引脚



3. 配置休眠唤醒引脚

Pioneer 套件上的开关 SW2 连接到一个休眠唤醒引脚 P0 [4]，并在按下时将端口引脚拉低。 要将此引脚配置为休眠唤醒开关，必须将其配置为电阻上拉。

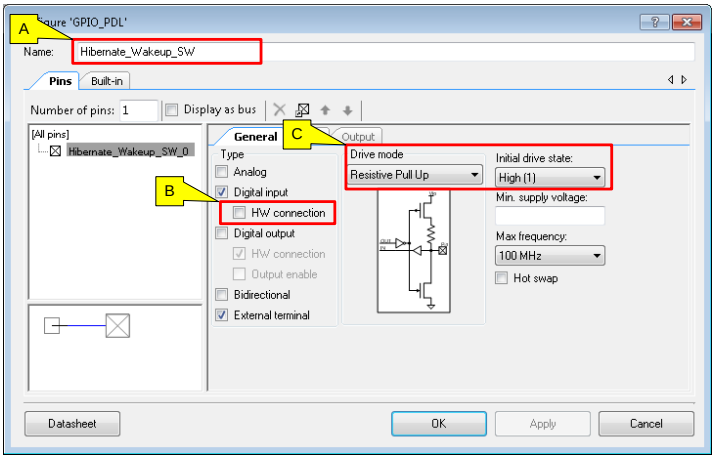
Figure 25 显示了配置。 双击组件打开配置对话框，然后执行以下操作：

A. 将组件实例的名称更改为 **Hibernate_Wakeup_SW**。

B. 取消选择 **HW connection(**硬件连接)。 固件应用程序固件不驱动此引脚。 但是，该引脚硬连线到休眠系统。 在固件中，唤醒将被配置为低电平有效。

C. 将 **Drive mode(**驱动模式)设置为 **Resistive Pull Up(**电阻上拉)和 **Initial drive state(**初始驱动状态)为 **High (1)** 高电平（1）。 这将配置器件检测从高电平到低电平的转换并将器件从休眠状态唤醒。

Figure 25. 配置输入引脚组件



4. 配置 UART 组件

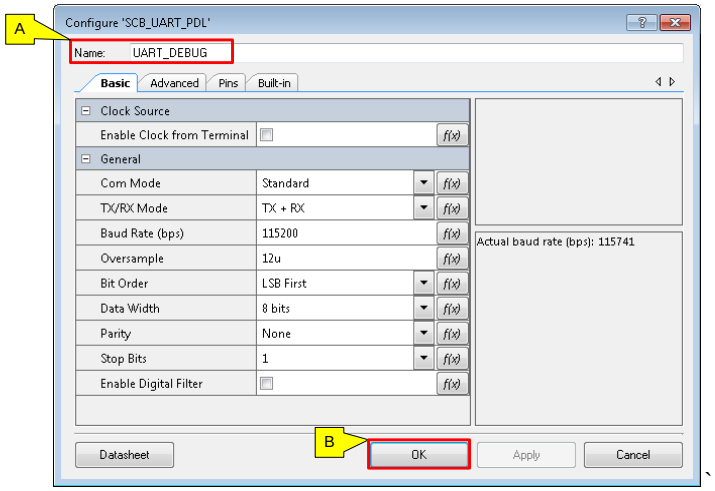
双击组件打开配置窗口。 该设计使用此组件在 115200 bps 的波特率下在终端窗口中显示调试消息。 这与 BLE 功能无关。

A. 将组件实例的名称更改为 **UART_DEBUG**。

B. 点击 **OK**.

对于其它设置，该设计使用默认值。

Figure 26. 配置基于 SCB 的 UART 组件



5. 设置 BLE 一般选项

双击原理图上的 BLE 组件打开配置窗口。设置 General 属性，如 Figure 27 所示。除了组件名称和堆栈操作外，此应用程序使用默认的常规属性。

A. 将名称更改为 BLE。

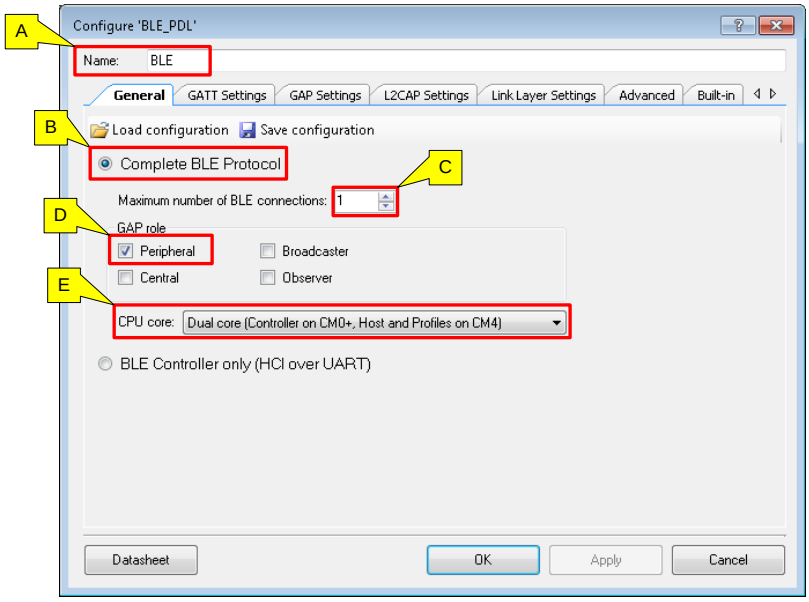
B. 确认选择了完整的 BLE 协议。

C. 将 BLE 连接的最大数量更改为 1.这将有利于正确配置 BLE 堆栈。

D. 确认选择 Peripheral 作为 GAP 角色。这将设备设置为 BLE 外围设备，并响应中央设备请求。

E. 对于 CPU 内核，选择双核（CM0 +上的控制器，CM4 上的主机和配置文件）选项。这将拆分 BLE 堆栈以在两个内核上工作。 CM0 +内核运行堆栈的 BLE 控制器部分，负责维护 BLE 连接。 BLE 主机在 CM4 内核上运行并执行应用程序级任务。这种双核心设置的主要优点是，当没有 BLE 相关任务未决时，CM4 内核可以进入深度休眠低功耗模式。

Figure 27. BLE 组件通用配置



6. 指定通用属性（GATT）设置

在这一步中，您将设置 BLE 配置文件，如 Figure 28 所示。

A. 点击 **GATT Settings** (GATT 设置)标签显示 GATT 选项。

B. Click the **Add Profile** drop-down menu. 点击添加配置文件下拉菜单。

C. 选择 **Find Me** > **Find Me Target (GATT Server)**选项。 这设置了 GAP 外设角色配置文件。 当菜单消失时，注意会出现一个新的服务“Immediate Alert”，如 Figure 29 所示。

Figure 28. BLE 组件 GATT 设置

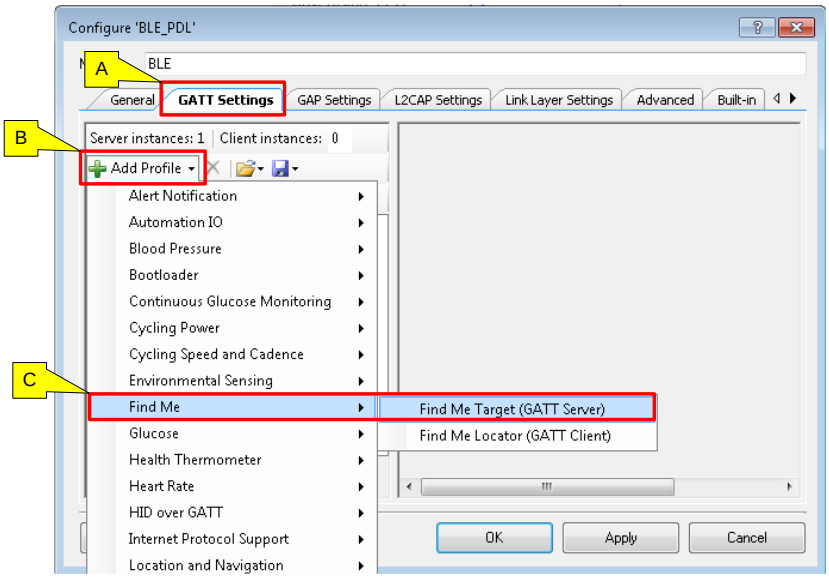
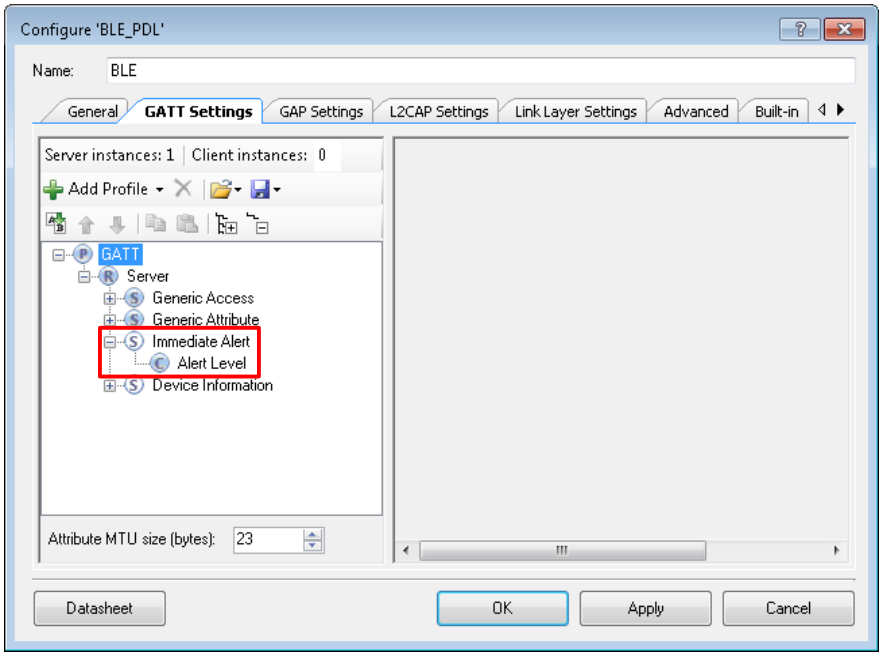


Figure 29. BLE 组件 Immediate Alert 服务添加



注意： 代码示例在 GATT 设置中也添加了器件信息服务以确认器件并读取固件版本。此服务对于 **Immediate Alert** 正常工作不是必须的。

7. 指定通用访问配置文件（GAP）常规设置

有一系列面板可以涵盖 GAP 设置。 通过左侧菜单能访问所有面板。 参见 Figure 30。

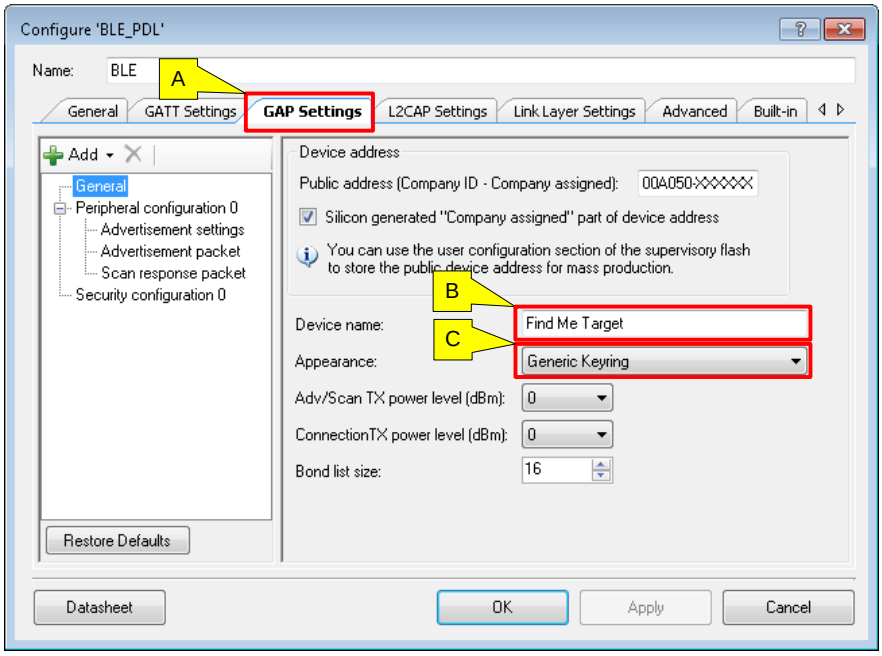
A. 单击 GAP 设置选项卡以显示 GAP 选项。 常规面板默认会出现。

B. 输入器件名称。设置为“**Find Me Target**”。

C. 将外观设置为通用密钥环。.

所有其他常规设置使用默认值。 这包括设备使用芯片生成的设备地址的“公司分配”部分。 这将配置主机设备尝试发现设备时显示的设备名称和类型，然后为设备分配唯一的 BLE 设备地址。

Figure 30. BLE 组件通用 GAP 设置



8. 指定 GAP 广告设置

请参阅 Figure 31 以获取此步骤的帮助。

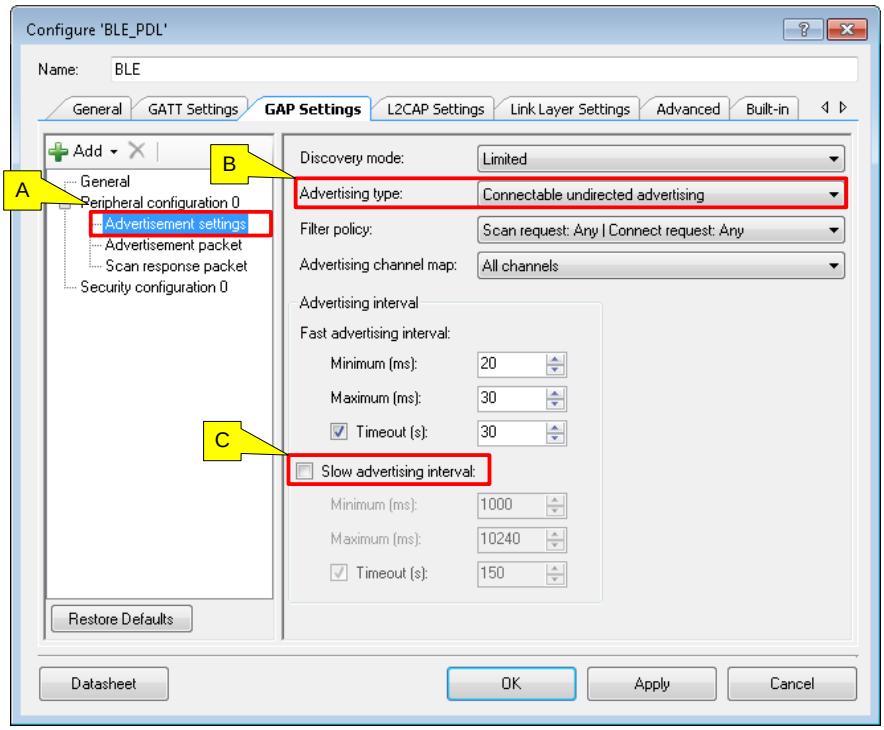
A. 点击左侧菜单内的**广告设置**项。面板出现。

B. 将**广告类型**设置为**可连接非定向广告**。

C. 取消选择**慢速广告时间间隔**复选框。

除此之外，默认值适用于此应用程序。 它使用有限的发现模式，广告超时时间为 30 秒，快速广告时间间隔为 20 到 30 毫秒。 快速广告允许快速发现和连接，但由于增加了 RF 广告数据包而消耗更多功率。

Figure 31. BLE 组件 GAP 广告设置



9. 指定 GAP 广告数据包设置

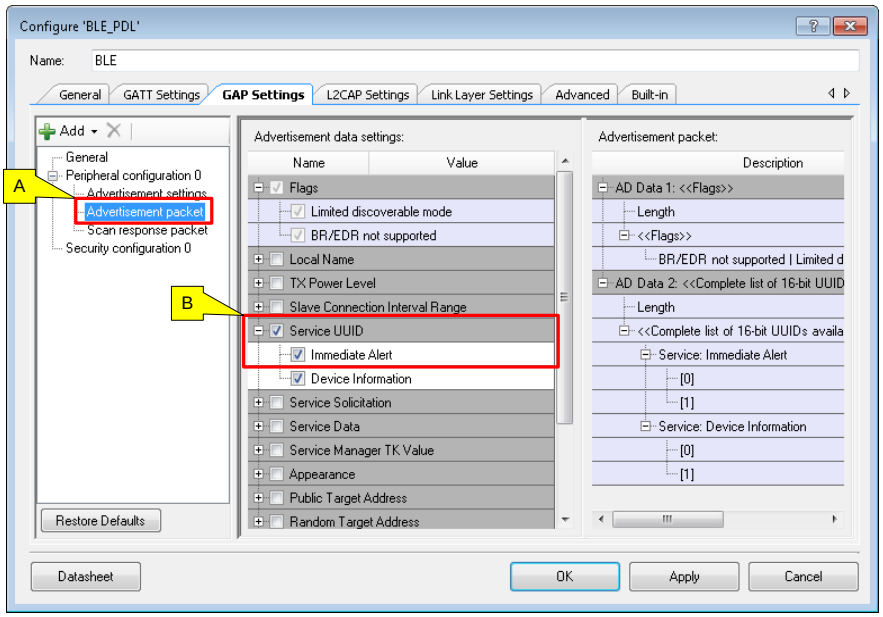
在此步骤中，您启用立即警报服务（IAS）的设备。 见图 Figure 32.

A. 点击左侧菜单中的**广告数据包**项。 面板出现。

B. 展开服务 UUID 项，然后选择即时警报。

这配置设备通知 BLE 中央设备 IAS 可用。 在添加项目时，广告数据包的结构和内容显示在配置面板的右侧。

Figure 32. BLE 组件 GAP 广告数据包设置



10. 指定扫描响应数据包设置

在此步骤中，指定扫描响应数据包的配置。 Figure 33 显示了结果。 请注意，在添加值时，扫描响应数据包的结构和内容将显示在配置面板的右侧。

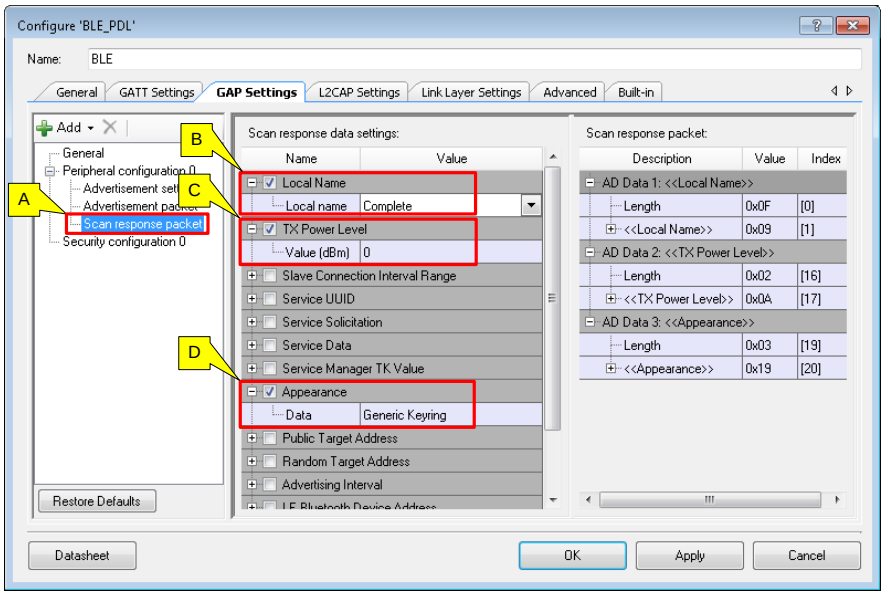
A. 点击左侧菜单中的扫描响应包项。 面板出现。

B. 选择本地名称以将该项目包含在响应中。 完成的默认设置为 OK。

C. 选择 **TX Power Level**（发射功率电平）以在数据包中包含该项目。

D. 选择外观以将该项目包含在数据包中

Figure 33. BLE 组件 GAP 扫描响应数据包



11. 指定安全配置设置

在此步骤中，您将为设备配置安全设置。 它使用不需要认证，加密或数据交换授权的配置。 参见 Figure 34。

A. 单击左侧菜单中的安全配置 0 项。 面板出现。

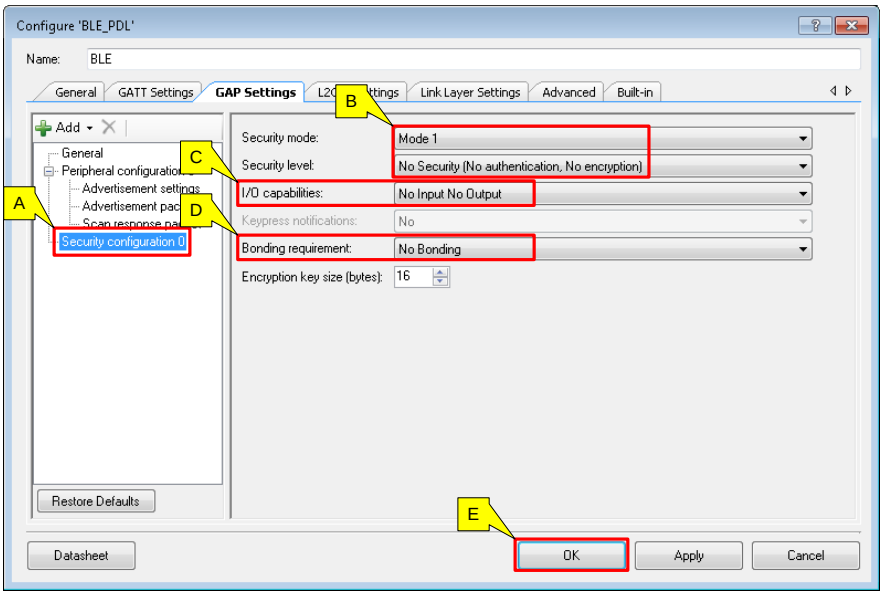
B. 确认安全模式是模式 1，安全级别是无加密。 如果不是，请修改设置。

C. 将 IO 功能设置为无输入无输出

D. 将绑定要求设置为不绑定。

E. 单击 **OK** 按钮完成 BLE 组件的配置。

Figure 34. BLE 组件 GAP 安全设置



单击 **OK** 以完成 BLE 组件配置。 在此设计中，所有其他设置都使用默认值，包括 LDCAP 设置、链路层设置和高级选项卡中的所有选项。

请参阅组件数据表以了解每种设置的意义。

12. 配置 MCWDT 组件以触发中断

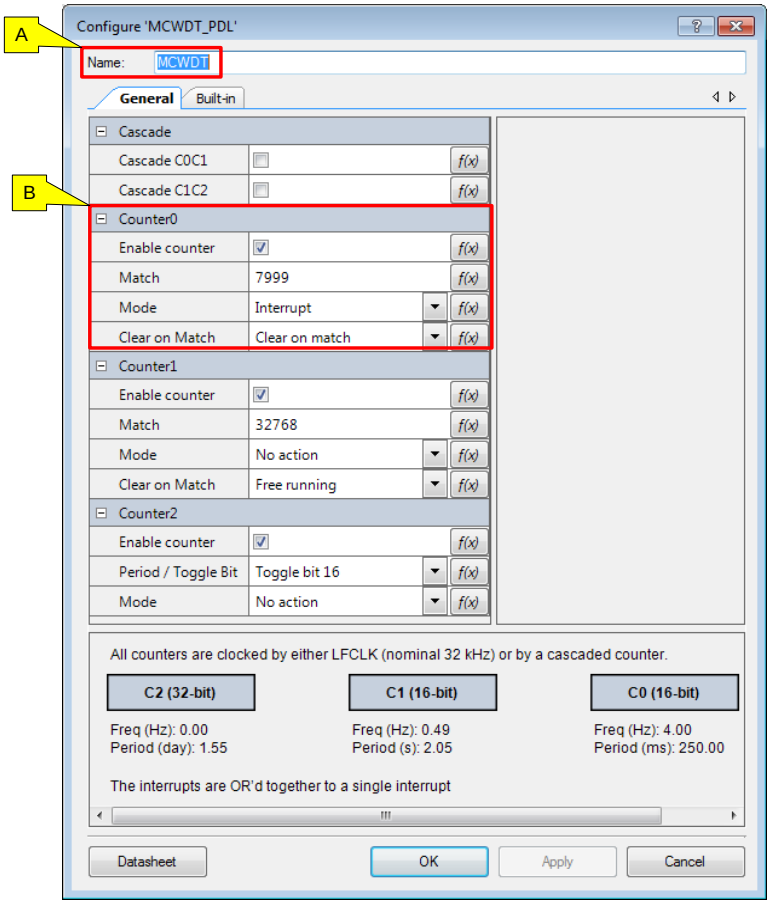
在此步骤中，将多计数器看门狗（MCWDT）组件配置为每 250 毫秒（4 Hz）触发一次中断。 MCWDT 的时钟源是低频时钟（LFCLK）。 LFCLK 的来源默认为内部低速振荡器（ILO）。 当收到 MILD 警报时，设计将使用此中断以 2 Hz 的频率闪烁警报 LED。 参见 Figure 35。

A. 将名称更改为 **MCWDT**。

B. 对于 **Counter0**，将匹配值更改为 7999，将模式更改为中断。 将 **Clear on Match** 字段设置为 **Clear on match**。

C. 单击 **OK** 按钮完成 MCWDT 组件的配置。

Figure 35. MCWDT_PDL 设置



13. 配置中断组件以在深度睡眠模式下唤醒 CM4 CPU

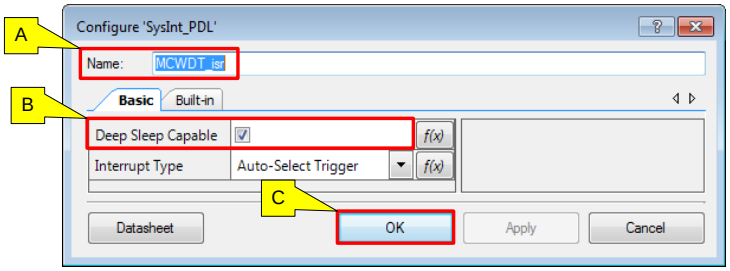
在这一步中，您将配置 SysInt 组件以唤醒 CM4 CPU。 参见 Figure 36。

A. 修改 **Name** 为 “**MCWDT_isr**”.

B. 选择中断为 **Deep Sleep Capable**.

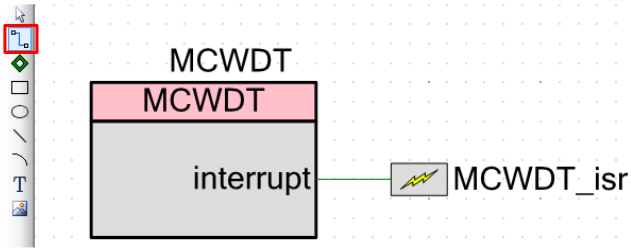
C. 点击 **OK** 按钮以完成系统组件的配置。

Figure 36. SysInt_PDL 设置



作为最后一步，将 MCWDT 组件的中断输出连接到 MCWDT_isr 组件输入。 这会将看门狗中断路由到 CPU（在稍后的步骤中，此中断的 CM4 CPU 的选择将在系统中断配置中进行设置）。 在原理图中，使用导线工具按钮或按下'W'键开始连线组件。

Figure 37. 连接 MCWDT 外设中断到 CM4 CPU



14. 为引脚组件设置物理引脚

完成设计仍有一项任务。 您必须将每个组件与设备上所需的物理引脚相关联。 选择使用哪个引脚由电路板设计驱动。 您可以在套件原理图中找到这些信息。 Figure 38 显示了此步骤的最终结果。 如果您使用自己的电路板或不带 LED 的开发套件，请选择适当的引脚。 您可以将外部 LED 连接到选定的引脚。

要设置引脚，请在相应的字段中键入端口号或引脚号，或使用下拉菜单选择端口或引脚。 通常使用端口号而不是引脚号，因为这些名称与正在使用的特定软件包无关。

A. 打开引脚选择器。

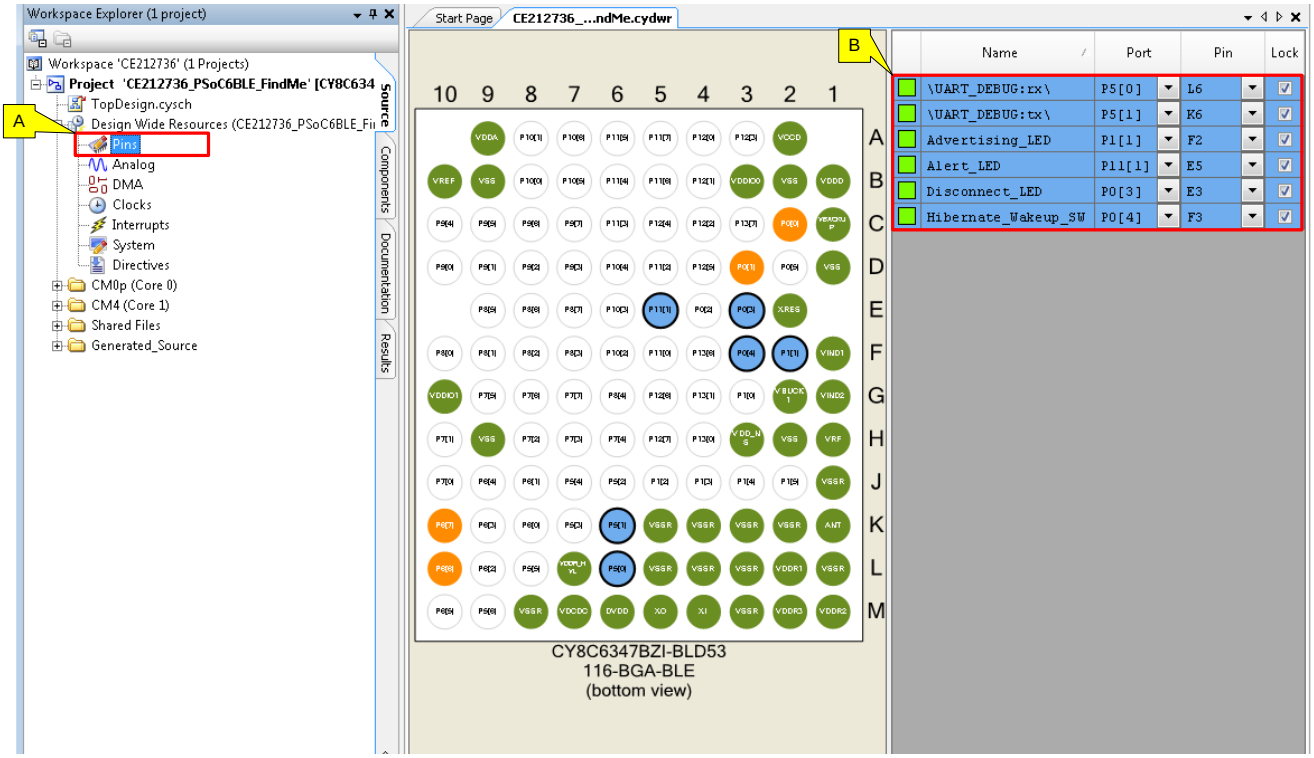
在 **Workspace Explorer** 窗格中，双击 Design Wide Resources 下的 Pins 项目。 此设备的引脚选择器出现。

B. 如 Table 1 所示设置每个引脚

Table 1. CY8CKIT-062-BLE 物理引脚分配

引脚组件名称	端口名称
UART_DEBUG:rx	P5[0]
UART_DEBUG:tx	P5[1]
Advertising_LED	P1[1]
Alert_LED	P11[1]
Disconnect_LED	P0[3]
Hibernate_Wakeup_SW	P0[4]

Figure 38. 引脚分配



15. 系统时钟配置

该设计使用高频系统时钟设置的默认值。虽然您不需要修改此设计的高频时钟，但您应该知道 PSoC Creator 如何管理它们。如果您正在使用自己的主板，则可能需要修改这些时钟。

在这一步中，您将低频时钟源设置为板上精确的时钟晶体振荡器（WCO）。该时钟由 BLE 子系统用于定时目的。

A. 在 Workspace Explorer 窗格中，双击 Design Wide Resources（CE212736.cydwr）下的 **Clocks** 项目。时钟列表出现。

B. 点击 **Edit Clock** 按钮，出现配置系统时钟对话框。

在这里你可以看到时钟树，并根据需要修改时钟。请注意，有不同类型的时钟的选项卡，例如源时钟，FLL / PLL，高频时钟和其他时钟。

C. 点击源时钟选项卡。

D. 通过选择 AltHF BLE ECO 模块中的复选框来启用 BLE ECO。参数应与电路板上使用的晶体相匹配。对于 CY8CKIT-062-BLE 套件，ECO 频率为 32 MHz，精度为±50 ppm。电路板上没有负载电容，因此使用 9.900 的最小规定负载电容（pF）。确保快速启动 ECO 晶体的启动时间为 785 微秒。

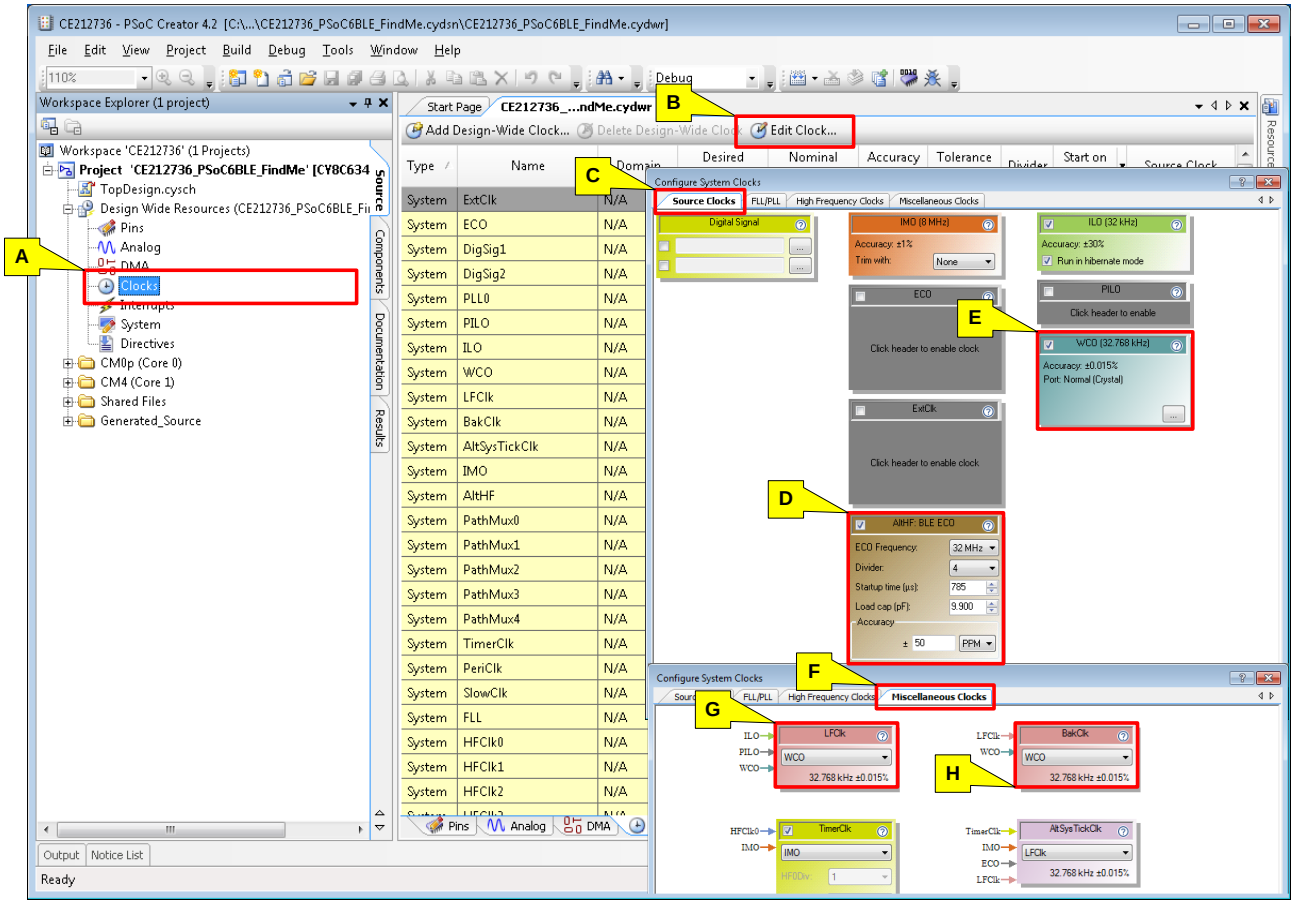
E. 模块中的复选框来启用 WCO 时钟。

F. 点击杂项时钟标签。

G. 选择 WCO 作为 LFClk 的源。

H. 选择 WCO 作为 BakClk 的源。参见 Figure 39。

Figure 39.时钟配置



16. 系统中断配置

在此步骤中，您将配置系统中断。 参见 Figure 40。

A. 在 Workspace Explorer 窗格中，双击 Design Wide Resources 下的 Interrupts 项目。 出现中断列表。

B. 确认为 Arm CM0 + CPU 启用了 BLE_bless_isr，优先级设置为 1，向量设置为 3. 这确保了 BLE 控制器中断由 CM0 +以最高优先级处理，而且在深度睡眠模式下。

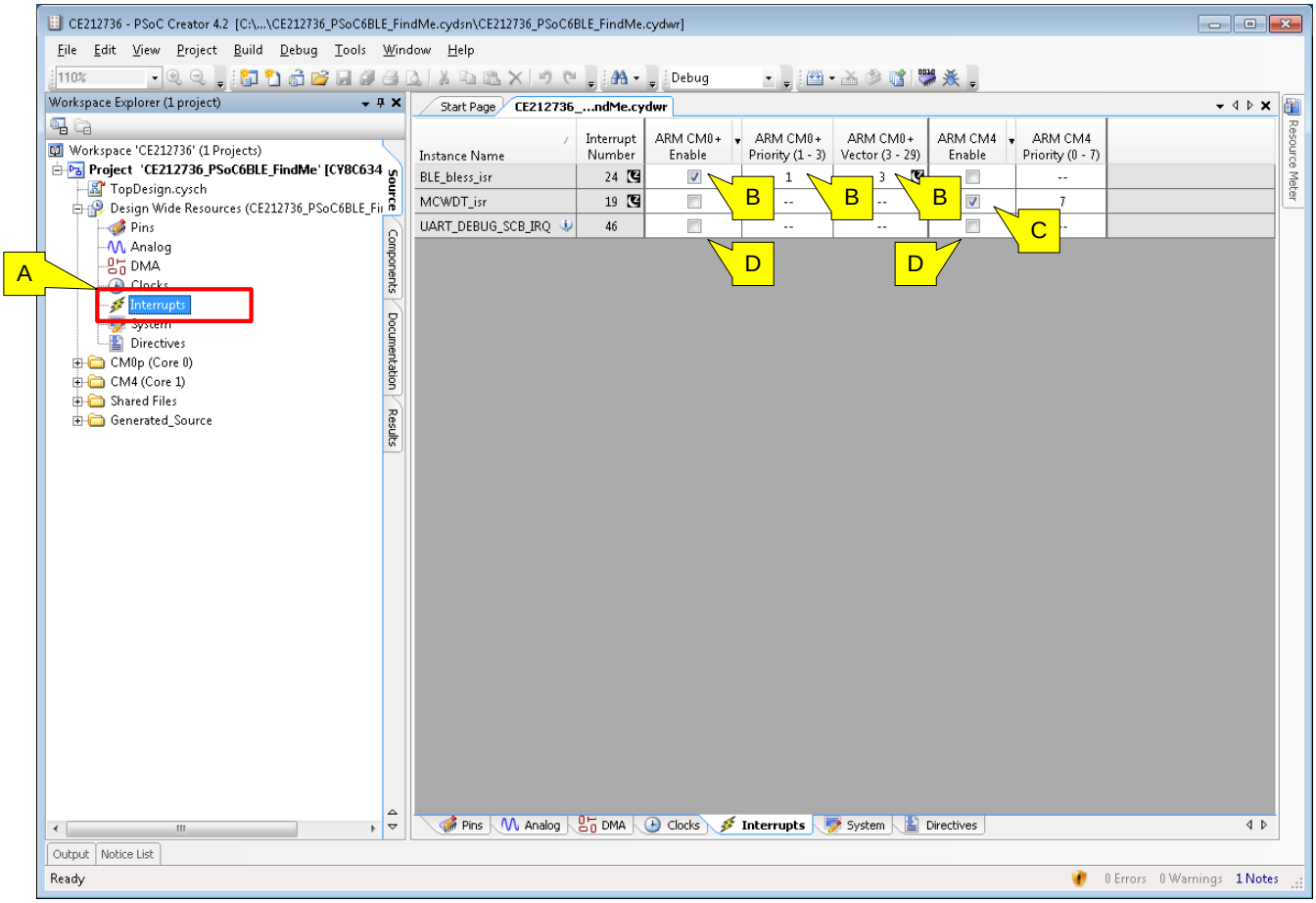
C. 为 Arm CM4 启用 MCWDT_isr。

D. 禁止两个内核的 UART_DEBUG_SCB_IRQ 中断。

取消选中相应的复选框。 您可以忽略与 UART_DEBUG 组件相关的中断，因为设计不使用它。

当您在 Part 3:中生成代码时，中断编号由 PSoC Creator 自动生成。

Figure 40.中断配置



开发过程的下一阶段是生成代码。

注意：本练习没有详细介绍如何将工作导出到目标 IDE。 但是，如果您希望使用目标 IDE，则在生成代码之前，这是工作流程中的一个要点，您应当确保选择了正确的目标 IDE。 请参阅支持其它 IDE。

6.6 Part 3:生成源代码

PSoC Creator 根据设计生成源代码。 推荐的工作流程是在编写固件之前生成代码。 PSoC Creator 将自动创建您可能会在固件中使用的宏、常量和 API 调用。

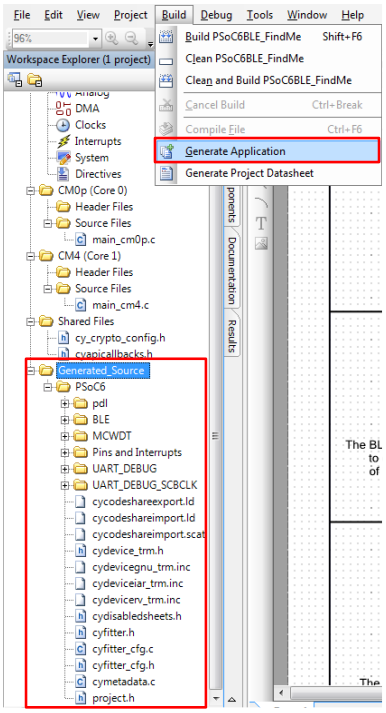
这部分练习非常简单，每个人的路径都是一样的。

路径	从零开始工作 代码示例仅供参考	PSoC Creator 或 BLE 新手使用代码示例	熟悉 PSoC Creator 和 BLE 时使用代码示例
操作	执行步骤 1	执行步骤 1.	执行步骤 1

1. 生成应用

选择 **Build > Generate Application。** PSoC Creator 根据设计生成源代码并将文件放入 *Generated_Source* 文件夹中。 请参见 Figure 41。 PSoC Creator 将提醒您可能发生的错误或问题。 如果您正在从头开始工作并遇到错误，请重新参阅第 2 部分：实施设计中的配置步骤，以确保您已正确执行它们。

Figure 41.生成应用

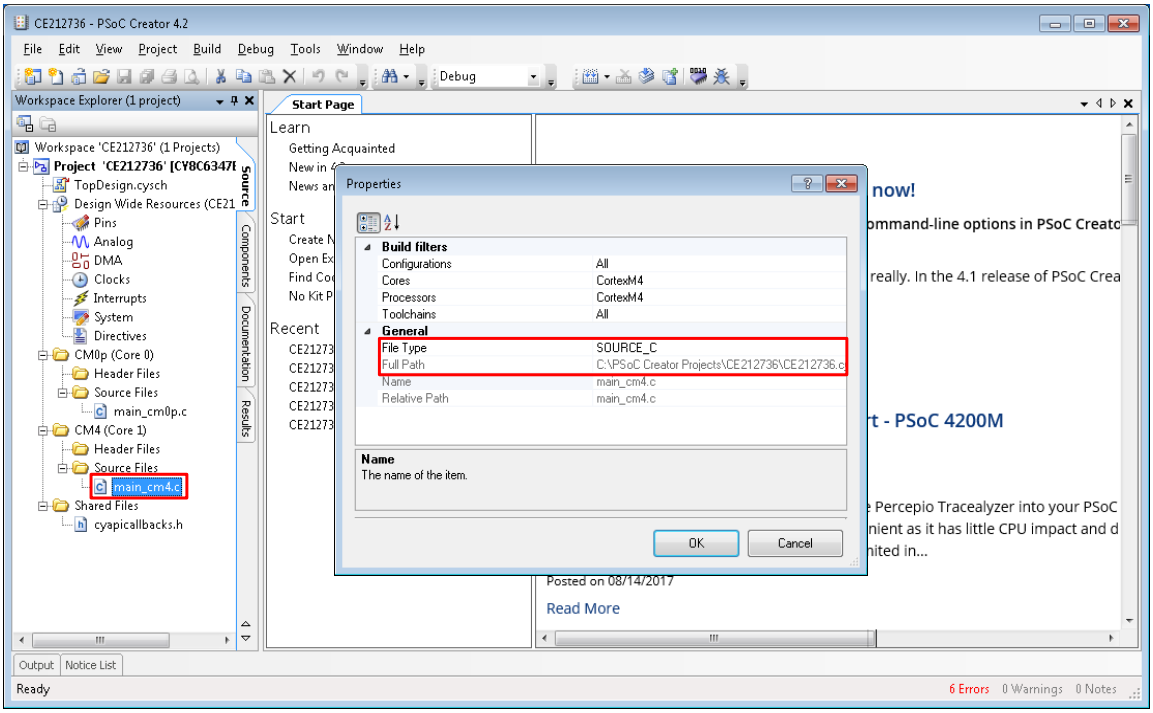


背景: PSoC 6 BLE 是一个双核平台。 您可以将固件定位到 Cortex-M4 或 Cortex-M0 +上运行。 您可以通过访问文件属性在源文件级别设置它。 右键单击源文件并选择属性。 Figure 42 显示了属性对话窗口。 此代码示例针对 Cortex-M4 内核。

默认情况下，main_cmp0p.c 文件针对 Cortex-M0 +，而 main_cm4.c 文件针对 Cortex-M4。 您不需要修改任何其他文件的属性。 它们已经在代码示例中设置。

按照惯例，CM0 +上运行的文件位于 CM0p 文件夹中，而 CM4 上运行的文件位于 CM4 文件夹中，但属性也必须适当设置—— 只将文件放入正确的文件夹中不会 导致它在特定的核心上运行。

Figure 42.设置源 C 文件的目标处理器



6.7 Part 4:编写固件

在开发过程的此阶段，您已经创建了一个项目，实现了硬件设计，并生成了代码。

在这一部分，你需要检查在应用中执行 BLE 功能的固件。

固件必须完成几项任务才能实施 BLE 标准配置文件应用程序，其中包括：

- 系统初始化
- 执行 BLE 堆栈事件处理程序
- 特定服务事件处理程序
- 提供主循环
- 执行低功耗操作（可选）

本部分的步骤讨论在 Part 2: 中配置的设计固件。 这些步骤不会检查每一行代码，而是指出实现重要功能的代码中的重要元素。

路径	从零开始工作 代码示例仅供参考	PSoC Creator 或 BLE 新手使用代码示例	熟悉 PSoC Creator 和 BLE 时使用代码示例
操作	执行所有步骤	跳转到步骤 1. 执行所有其它步骤。	如果愿意，可以跳过这个部分。跳转到 Part 5: 编译项目并对设备编程

代码示例具有所有必需的代码。 如果您正在从头开始工作，则在第一步中从代码示例项目复制源文件。 如果您正使用代码示例，那些文件已存在于您的项目中，因此您可以跳过第一步。

1. 添加文件到项目中

如果您使用的是代码示例，则可以跳过此步骤。 代码示例已经有了所需的源文件。

如果您正在从头开始工作，则所需的源代码文件不在您的项目中。

A. 找到 CE212736.cydsn 文件夹，其中包含代码示例的源文件。 该文件夹位于您之前下载的代码示例工作区归档文件中。

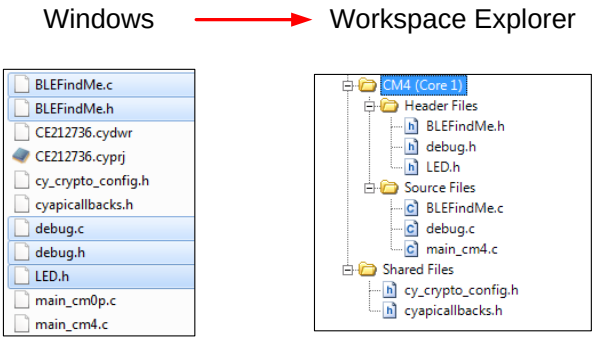
B. B.将这些文件从 CE212736.cydsn 文件夹复制到项目的.cydsn 文件夹中。 替换任何现有的文件。

- BLEFindMe.h
- debug.h
- LED.h
- BLEFindMe.c
- debug.c
- main_cm0p.c
- main_cm4.c

C. 将这些文件添加到您的项目。 您可以将它们从 Windows 文件夹拖到 Workspace 浏览器中，然后将它们放入工作区中的 CM4 文件夹位置。 参见 Figure 43。

- BLEFindMe.h
- debug.h
- LED.h
- BLEFindMe.c
- debug.c

Figure 43.添加文件到项目



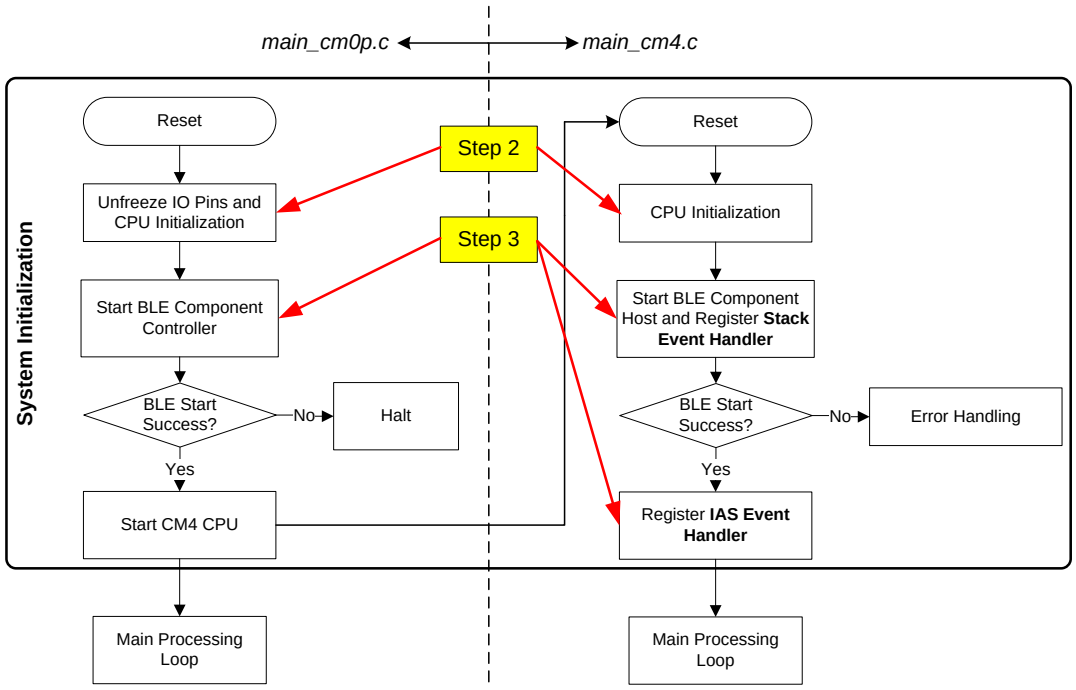
您不必添加 main_cm0p.c 或 main_cm4.c。 该项目默认具有这些文件。 您只需将它们替换到项目文件夹中，以便项目使用刚刚复制的新版本。

2. 系统初始化

在剩下的步骤中，我们检查 main_cm0p.c 和 main_cm4.c 文件中的代码。 为了清楚起见，代码片段经常会删除调试打印语句。 请参阅实际的源文件以全面了解代码。

Figure 44 显示了初始化系统的步骤。 当 PSoC 6 BLE 器件复位时，固件首先执行系统初始化，包括设置要执行的 CPU 内核，启用全局中断以及启用设计中使用的其他组件。 初始化在 CPU 内核之间分配。 CM0p CPU 出现复位并尝试启动 BLE 控制器部分。 如果成功，则 CM0p CPU 将启用 CM4 CPU。 CM4 CPU 将启动 BLE 主机部分并注册必要的应用程序端处理函数。

Figure 44. 系统初始化流程图 – CM4



main_cm0p.c 中的代码声明了一个局部变量来保存来自 BLE API 调用的返回值。 关键任务是启用 BLE 控制器，并设置 CM4 以使应用程序代码运行。 在主循环中，CM0p CPU 处理控制器上待处理的 BLE 事件。 在没有待处理事件的情况下，CM0p 进入深度睡眠模式。

```
int main(void)
```

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

       /* Cy_Ble_ProcessEvents() allows BLE stack to process
          pending events */

       /* The BLE Controller automatically wakes up host if required */
       Cy_BLE_ProcessEvents();
    }
```

main_cm4.c 中的代码初始化要由 CM4 使用的关键组件，并持续运行 BLE 应用程序进程。 BleFindMe_Init（）例程初始化并启动所有组件，包括设置 CM4 中断。 它执行启用 BLE 主机的关键任务。

在应用程序过程中，CM4 CPU 会处理主机上挂起的 BLE 事件。 如果没有待处理的事件，CM4 进入深度睡眠低功耗模式。

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

   /* Initialize BLE */
   BleFindMe_Init();

   for(;;)
```

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

3. 启动 BLE 组件和注册事件处理程序

CPU 初始化后，固件会初始化 BLE 组件，该组件会设置完整的 BLE 子系统。 BleFindMe_Init（）子例程处理工作。要聚焦关键任务，调试打印语句已被删除。检查源文件以查看完整的代码。

作为 BLE 组件初始化的一部分，您必须传递事件处理函数，该函数将由 BLE 堆栈调用以通知未决事件。如果 BLE 组件初始化成功，则固件为特定于 IAS 的事件注册第二个事件处理程序。

该代码使用 PDL API 函数调用来配置应用程序。首先它启动 BLE 组件。该参数是堆栈事件处理函数的地址。

该代码也获得堆栈版本。如果调试端口已启用，则版本将打印在串行通讯窗口中。

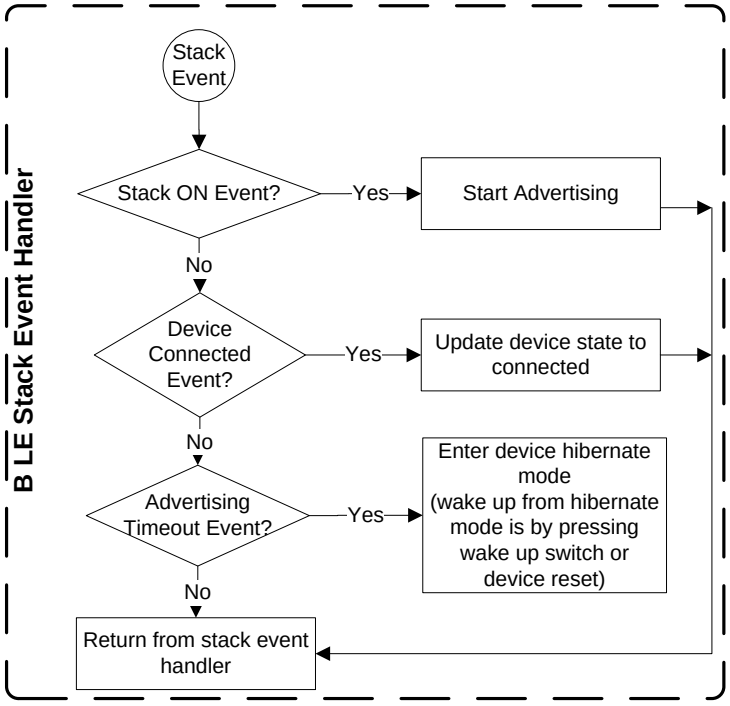
然后注册 IAS 事件处理程序以处理即时警报服务相关事件。最后，它配置并使 MCWDT 每 250 毫秒触发一次中断。

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

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

     /* Start BLE component and register generic event handler */
    apiResult = Cy_BLE_Start(StackEventHandler);

    apiResult = Cy_BLE_GetStackLibraryVersion(&stackVersion);

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

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

       /* Configure ISR connected to MCWDT interrupt signal*/
       /* MCWDT_isr_cfg structure is defined by the SYSINT_PDL component based on
          parameters entered in the customizer. */
       Cy_SysInt_Init(&MCWDT_isr_cfg, &MCWDT_Interrupt_Handler);
       /* Clear CM4 NVIC pending interrupt for MCWDT */
       NVIC_ClearPendingIRQ(MCWDT_isr_cfg.intrSrc);
       /* Enable CM4 NVIC MCWDT interrupt */
    NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc);

}
```

4. 执行堆栈事件处理程序

BLE 组件内的 BLE 堆栈生成事件。 这些事件通过 BLE 堆栈事件处理程序向应用程序固件提供状态和数据。 Figure 45 显示了表示某些事件的简化流程图

Figure 45. BLE 堆栈事件处理程序流程图



事件处理程序必须处理堆栈中的几个基本事件。 对于此代码示例中的 Find Me Target 应用程序，BLE 堆栈事件处理程序必须处理 Table 2.中描述的事件。实际代码可识别并响应其他事件，但它们对于此应用程序不是强制性的。

Table 2. BLE 堆栈事件

BLE 堆栈事件名称	事件描述	事件处理程序操作
CY_BLE_EVT_STACK_ON	BLE 堆栈初始化已成功完成	开始广告并在 LED 上反映广告状态
CY_BLE_EVT_GAP_DEVICE_DISCONNECTED	与对等设备的 BLE 链路断开	重新启动广告并在 LED 上反映广告状态
CY_BLE_EVT_GAP_DEVICE_CONNECTED	建立与对端设备的 BLE 链接	在 LED 上更新 BLE 链接状态
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP	BLE 堆栈广告开始/停止事件	关闭 BLE 堆栈
CY_BLE_EVT_HARDWARE_ERROR	BLE 硬件错误	更新 LED 状态以反映硬件错误并暂停 CPU
CY_BLE_EVT_STACK_SHUTDOWN_COMPLETE	BLE 堆栈已关闭	将器件配置为休眠模式并等待唤醒引脚上的事件

代码片段显示了事件处理程序如何响应已识别事件的两个示例。 请参阅实际的源代码以获得完整的理解。

在这段代码中，处理程序响应“广告开始/停止”事件。 该代码适当地切换 LED。 如果广告已经开始，则广告 LED 亮起。 断开 LED 指示灯熄灭，因为设备开始发布广告并准备好进行连接。 如果广告停止，代码会正确设置 LED，并设置一个标志进入休眠模式。

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

```
        Cy_GPIO_Write(Disconnect_LED_0_PORT, Disconnect_LED_0_NUM, LED_OFF);
      }
      else if(Cy_BLE_GetAdvertisementState() == CY_BLE_ADV_STATE_STOPPED)
      {
        DEBUG_PRINTF("Advertisement stopped \r\n");
        Cy_GPIO_Write(Advertising_LED_0_PORT, Advertising_LED_0_NUM, LED_OFF);
        Cy_GPIO_Write(Disconnect_LED_0_PORT, Disconnect_LED_0_NUM, LED_ON);

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

在这段代码中，处理程序响应“断开”事件。 它正确设置 LED，并设置休眠标志。

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

  case CY_BLE_EVT_GAP_DEVICE_DISCONNECTED:
       if(Cy_BLE_GetConnectionState(appConnHandle) == CY_BLE_CONN_STATE_DISCONNECTED)
       {
         DEBUG_PRINTF("CY_BLE_EVT_GAP_DEVICE_DISCONNECTED %d\r\n",
CY_BLE_CONN_STATE_DISCONNECTED);
         alertLevel = CY_BLE_NO_ALERT;

        Cy_GPIO_Write(Advertising_LED_0_PORT, Advertising_LED_0_NUM, LED_OFF);
        Cy_GPIO_Write(Disconnect_LED_0_PORT, Disconnect_LED_0_NUM, LED_ON);

        /* Enter into discoverable mode so that remote device can search it */
        apiResult = Cy_BLE_GAPP_StartAdvertisement(CY_BLE_ADVERTISING_FAST,
CY_BLE_PERIPHERAL_CONFIGURATION_0_INDEX);

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

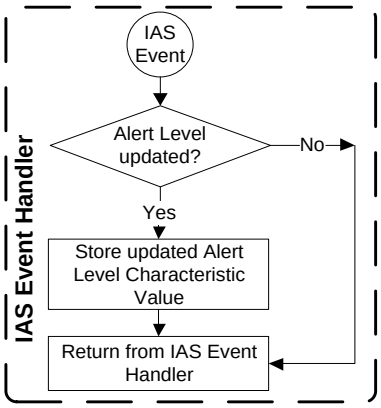
这些片段让您了解事件处理程序如何响应事件。 检查实际功能以查看每个事件的处理方式。

5. 执行业务特定事件处理程序

BLE 组件还会生成与设计支持的每种服务相对应的事件。 对于 Find Me Target 应用程序，BLE 组件会生成 IAS 事件，让应用程序知道警报级别特性已更新为新值。 事件处理程序获取新值并将其存储在变量 alertLevel 中。 主回路根据当前警报级别切换警报 LED。

Figure 46 展示了 IAS 事件处理流程图

Figure 46. BLE IAS 事件处理流程图



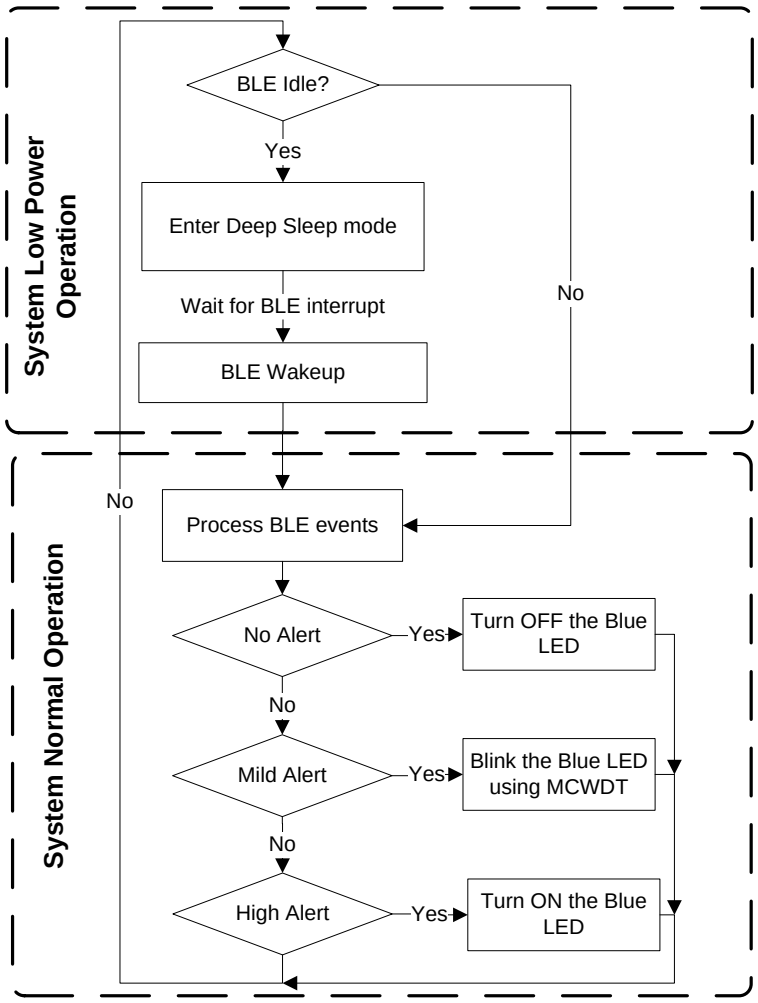
该代码片段显示了固件如何完成此任务。

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

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

6. 事件发生时的处理（主环路）

主环路仅仅调用 `BleFindMe_Process()`。Figure 47 显示 `BleFindMe_Process()` 流程图。

Figure 47. 固件主环路流程图



如果没有未决的 BLE 主机事件并且没有主动交互，则应用程序进入低功耗模式。 然后它会通知 BLE 处理事件，并根据警报级别更新 LED。

```
        /* The call to EnterLowPowerMode also causes the device to enter hibernate
       mode if the BLE is disconnected. */
    EnterLowPowerMode();
    /* Cy_Ble_ProcessEvents() allows BLE stack to process pending events */
    Cy_BLE_ProcessEvents();

    /* Update Alert Level value on the Blue LED */
    switch(alertLevel)
    {
        case CY_BLE_NO_ALERT:
            /* Disable MCWDT interrupt at NVIC */
            NVIC_DisableIRQ(MCWDT_isr_cfg.intrSrc);
            /* Turn the Blue LED OFF in case of no alert */
            Cy_GPIO_Write(Alert_LED_0, LED_OFF);
            break;

        /* Use the MCWDT to blink the Blue LED in case of mild alert */
        case CY_BLE_MILD_ALERT:
            /* Enable MCWDT interrupt at NVIC */
```

```
            NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc);
            /* The MCWDT interrupt handler will take care of LED blinking */
            break;

        case CY_BLE_HIGH_ALERT:
            /* Disable MCWDT interrupt at NVIC */
            NVIC_DisableIRQ(MCWDT_isr_cfg.intrSrc);
            /* Turn the Blue LED ON in case of high alert */
            Cy_GPIO_Write(Alert_LED_0, LED_ON);
            break;

        /* Do nothing in all other cases */
        default:
            break;
    }
```

这就完成了代码示例中固件工作原理的总结。 随意浏览源文件以获得更深入的了解。

6.8 Part 5: 编译项目并对设备编程

本节介绍如何对 PSoC 6 BLE 器件进行编程。 如果您正在使用带内置编程器的开发套件（例如 CY8CKIT-062-BLE Pioneer 套件），请使用 USB 电缆将开发板连接到计算机。 如果您正在自己的硬件上开发，则可能需要硬件编程器/调试器; 例如，赛普拉斯 CY8CKIT-002 MiniProg3。

路径	从零开始工作 代码示例仅供参考	PSoC Creator 或 BLE 新手使用代码示例	熟悉 PSoC Creator 和 BLE 时使用代码示例
操作	执行所有步骤		

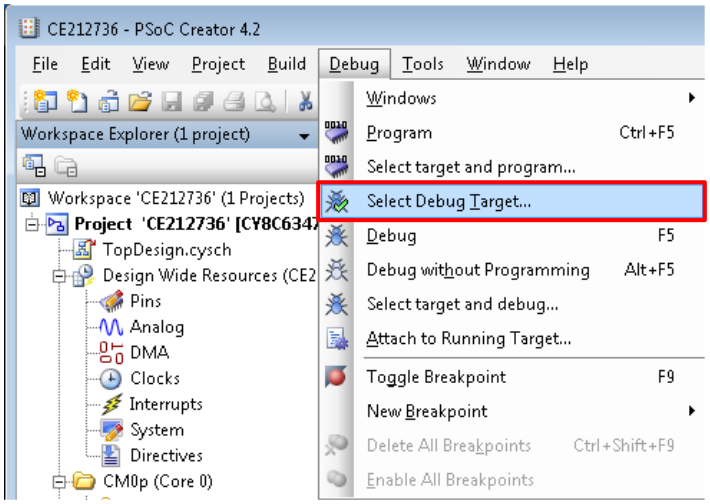
如果您正在从头开始工作并遇到错误，请重新阅读之前的步骤以确保您完成了所有必需的任务。 您可以解决错误或切换到这些最终步骤的代码示例。

1. 选择调试目标

PSoC Creator 每次可以调试一个内核。

A. 在 PSoC Creator 中, 选择 **Debug** > **Select Debug Target**, 如 Figure 48 所示。

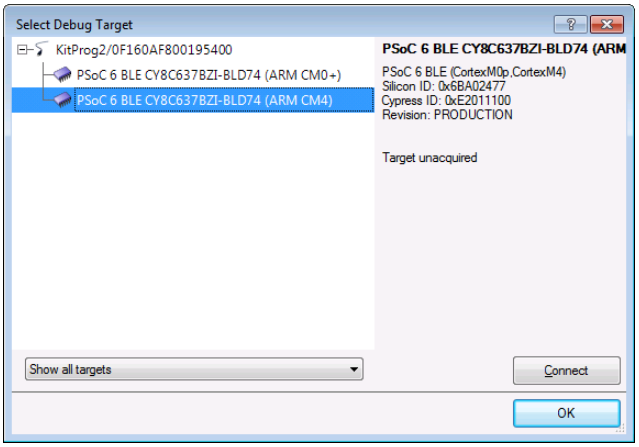
Figure 48. 选择调试目标



B. 连接器件板

在 **Select Debug Target** 对话框，选择 CM4 目标，然后点击 **OK/Connect**, 如 Figure 49 所示。

Figure 49.连接到器件

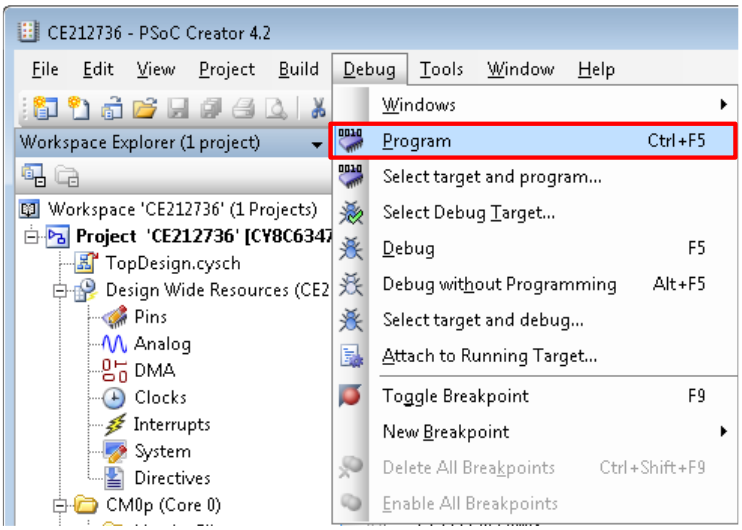


提示：对于编程电路板，您可以选择任一目标。 内核共享相同的内存空间。 编程两个内核程序中的任意一个都会编程两个内核程序。 但是，如果你正在调试这个选择很重要。 调试器将只能看到您连接的内核。 这些说明不使用调试器。

2. 编程器件板

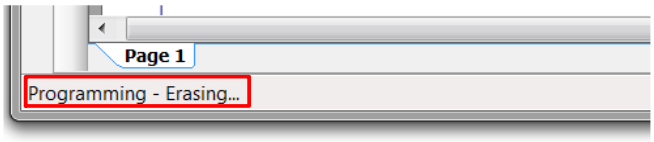
选择 **Debug** > **Program** 对项目的器件编程，如 Figure 50 所示。

Figure 50. 对器件编程



您可以在窗口 PSoC Creator 窗口的左下角查看编程状态，如 Figure 51 所示。

Figure 51.编程状态



在双核应用程序中，链接程序文件将每个可执行文件放在内存中的正确位置。 CM0 +内核开始执行，启用 CM4 内核。

编程完成后，应用程序运行。 LED 变成绿色，表示目标正进行广告。 发生广告超时后，它会变成红色，表示它已断开连接。

提示：“调试”>“调试”命令也可以对器件板进行编程。 如果需要生成或重建任何代码，那么在发出编程或调试命令时会自动发生。 您也可以在不编程器件板的情况下进行调试。 但是，这些说明不使用调试器。

6.9 Part 6: 测试您的设计

本节介绍如何使用 CySmart 或 CySmart 测试 BLE 设计。 Figure 11 显示了使用 BLE Pioneer 套件测试设计的设置。

路径	从零开始工作 代码示例仅供参考	PSoC Creator 或 BLE 新手使用 代码示例	熟悉 PSoC Creator 和 BLE 时使用代码示例
操作	执行步骤 1 或步骤 2		

1. 使用 CySmart Mobile App 进行测试

A. 在 iOS 或 Android 设备上打开蓝牙。.

B. 启动 CySmart 应用程序。

C. 按下 BLE Pioneer 套件上的复位开关，从设计中启动 BLE 广告。 为使 CySmart 应用程序能够看到设备，绿色 LED 必须亮起。

D. 拉下 CySmart 应用程序主屏幕以开始扫描 BLE 外设。 Find Me 目标在 CySmart 应用主屏幕中显示为 BLE 设备。 点击以建立 BLE 连接。 如果手机没有找到目标，请再次按下重置按钮，或尝试使用 CySmart 主机仿真工具（步骤 2）。

E. 从旋转视图中选择“查找我”配置文件。

F. 在“查找我的配置文件”屏幕上选择一个 Alert Level 值，并根据您的选择查看设备上蓝色 LED 的状态。

Figure 52 显示了使用 iOS 应用程序的此过程。 Figure 53 显示了使用 Android 应用程序的过程。

Figure 52. 用 CySmart iOS App 测试

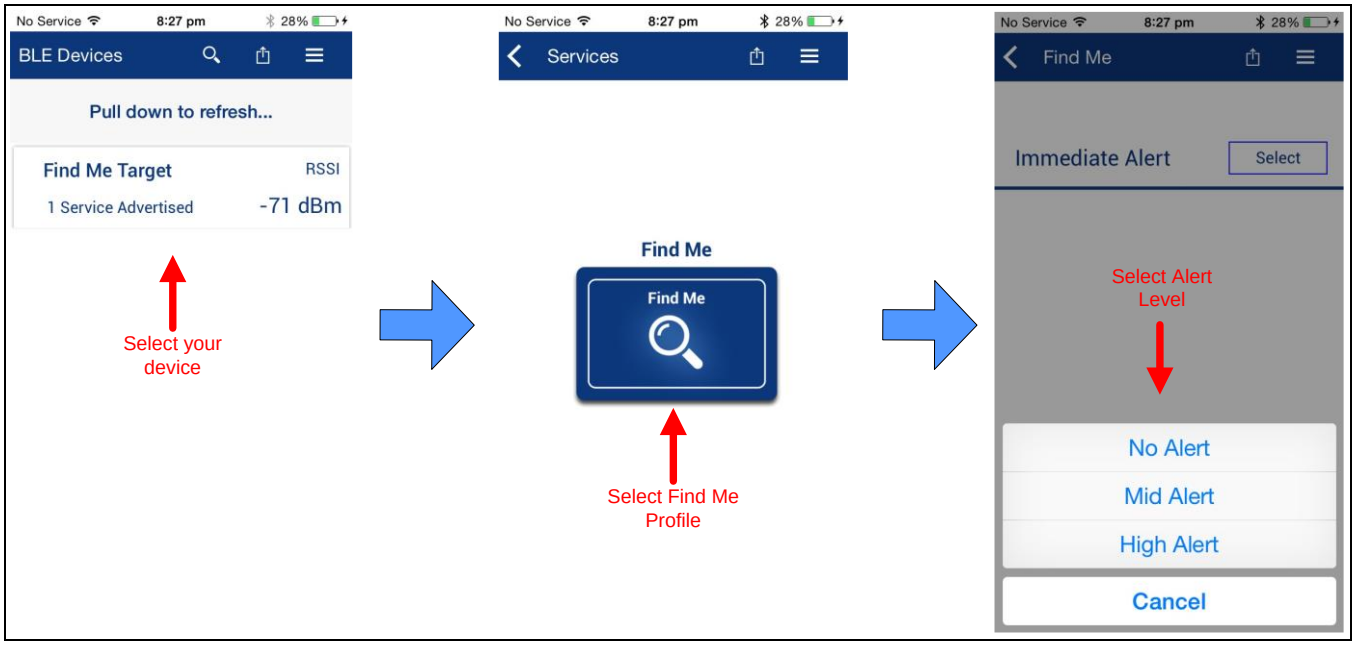
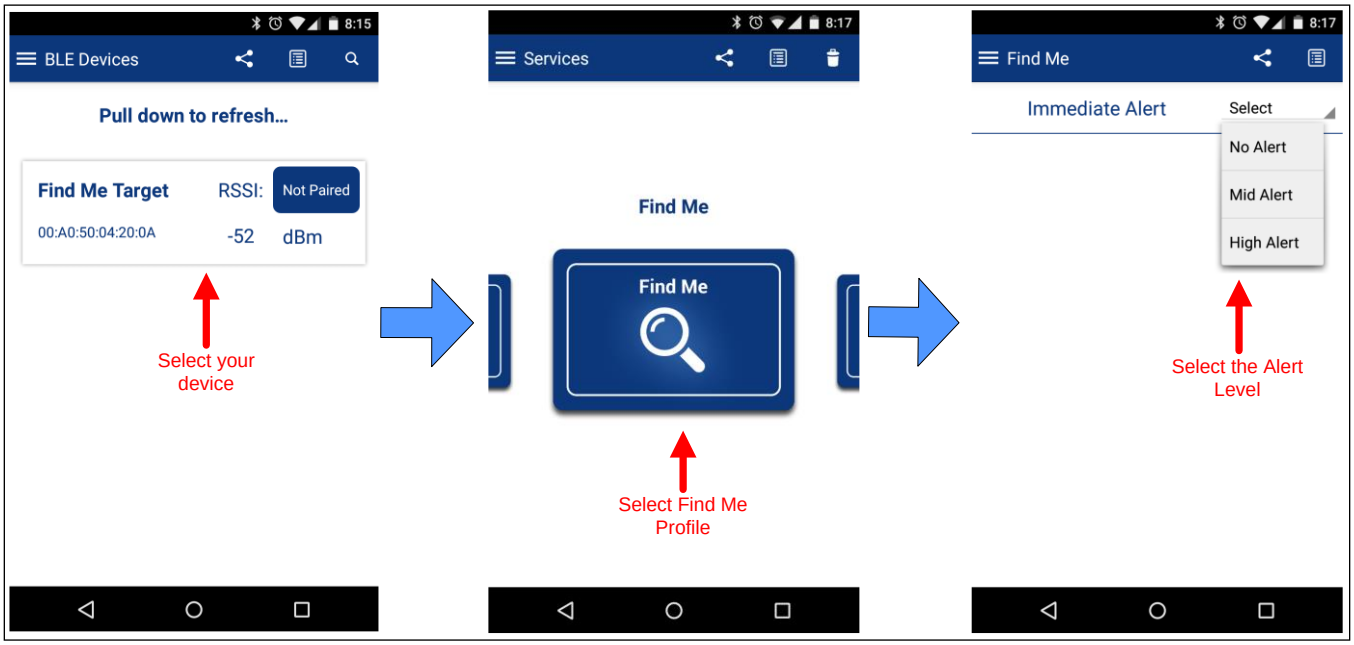


Figure 53. 用 CySmart Android App 测试



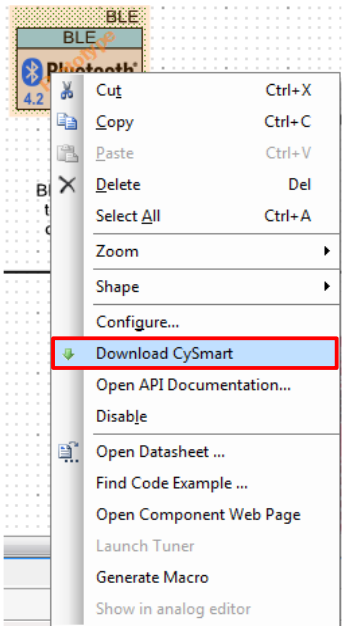
2. 使用 CySmart 主机仿真工具测试

作为 CySmart 移动应用的替代方案，您可以使用 CySmart 主机仿真工具与您的设计建立 BLE 连接，并对 BLE 特性执行读取或写入操作。

如果尚未安装，请安装 CY 智能主机仿真工具。

右键单击顶部原理图中的 BLE 组件符号，然后选择下载 CySmart，如 Figure 54 所示。按照安装程序指示安装该工具。

Figure 54. 下载 CySmart

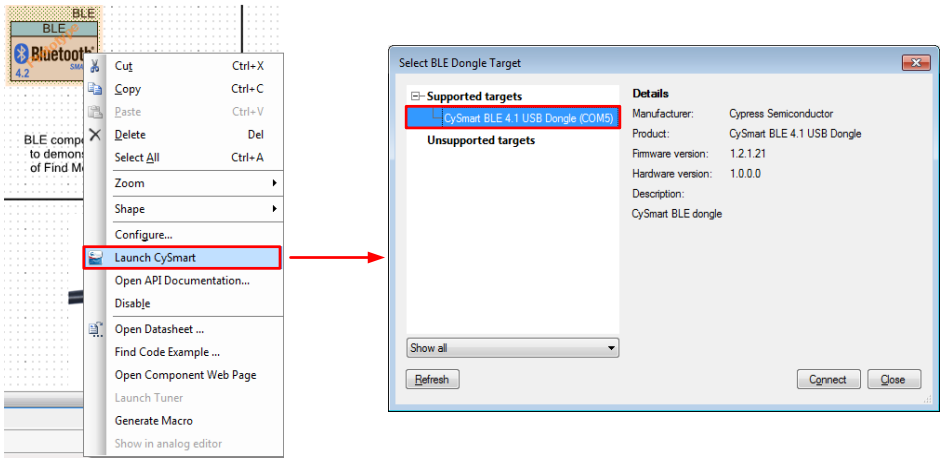


将 BLE 软件狗连接到 Windows 个人电脑。 等待驱动程序安装完成。

启动 CySmart 主机仿真工具。

该工具会自动检测 BLE 软件狗。 如果 BLE 加密狗没有出现在选择 BLE 加密狗目标弹出窗口中，请单击刷新。点击连接，如 Figure 55.所示。

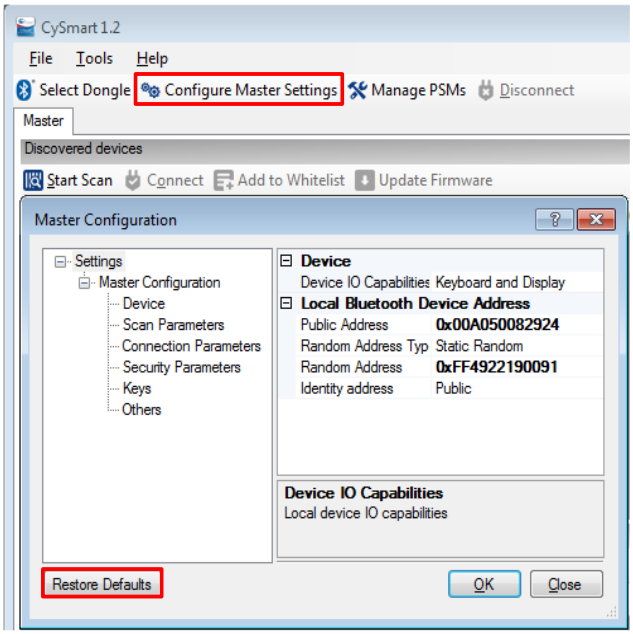
Figure 55. CySmart BLE 加密狗设置



注意：如果加密狗固件已过时，您将收到警报。 您必须先升级固件，然后才能完成此步骤。 按照窗口中的说明更新加密狗固件。

G. 点击 Configure Master Settings，然后点击 Restore Defaults，如 Figure 56 所示。然后点击 OK。

Figure 56. CySmart 主机设置配置

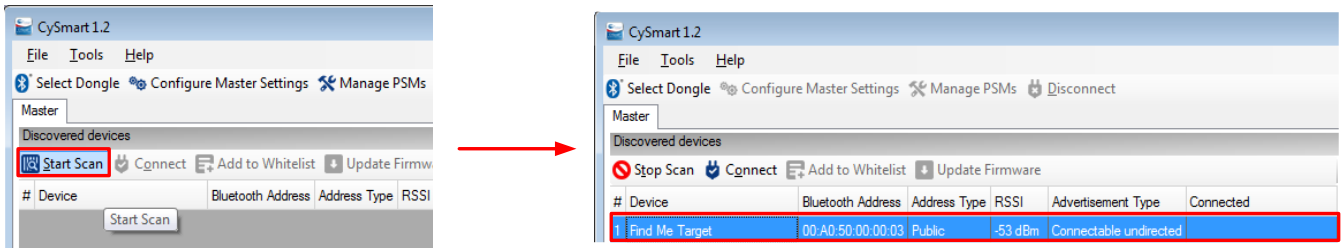


按下 BLE Pioneer 套件上的复位开关，从您的设计中启动 BLE 广告。 LED 变成绿色表示您正在进行广告。

在 CySmart 主机仿真工具上，单击开始扫描。 您的设备名称（配置为“查找我的目标”）应显示在“发现的设备”列表中，如 Figure 57 所示。

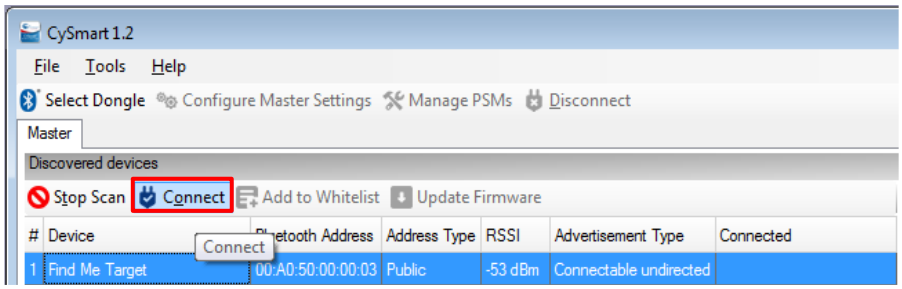
注意：如果扫描过程超时但未找到目标，则 LED 变为红色。 再次点击重置按钮以恢复广告。

Figure 57. CySmart Device Discovery



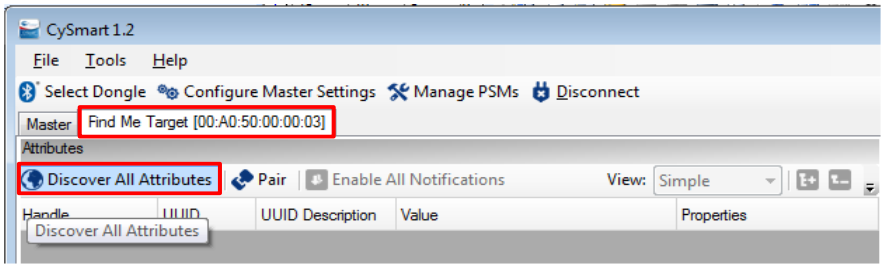
H. 选择 Find Me Target，然后单击 Connect 在 CySmart 主机仿真工具和您的设备之间建立 BLE 连接，如 Figure 58 所示。

Figure 58. CySmart 设备连接



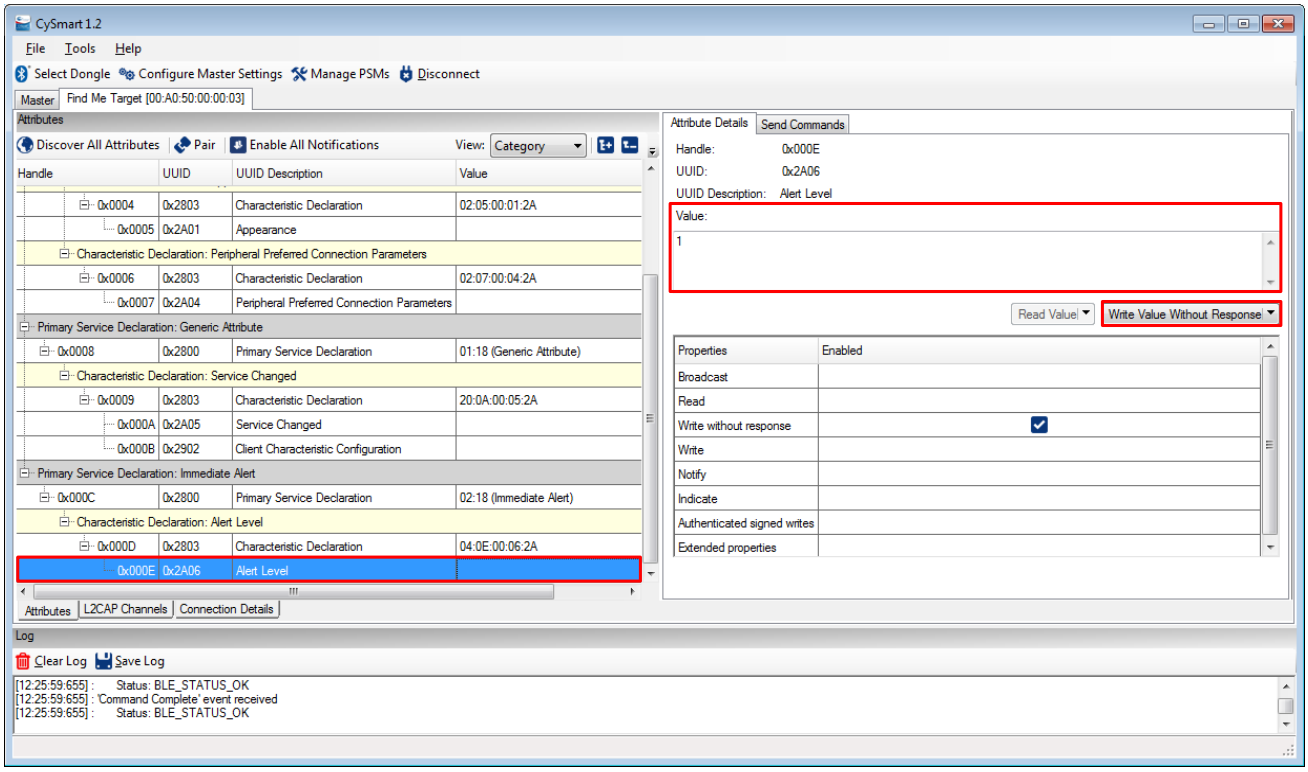
I. 连接后，切换到 Find Me Target 设备选项卡，并从 CySmart 主机仿真工具中发现设计中的所有属性，如 Figure 59 所示。

Figure 59. CySmart 属性发现



J. 向下滚动“属性”窗口并找到“立即警报服务”字段。 如 Figure 60 所示，在即时警报服务下向“警报级别”特征写入 0,1 或 2 的值。 根据您的警报级别特征配置，观察设备上 LED 的状态变化。

Figure 60.以 Cysmart 主机仿真工具测试



完成探索后，单击 CySmart 应用程序中的断开按钮。 LED 变成红色表示您已断开连接。

您可以通过按下电路板上的重置按钮来重复此过程以恢复广告。

7 总结

本应用笔记介绍了 BLE 协议、PSoC 6 BLE 器件架构和开发工具的基础知识。PSoC 6 BLE 是真正的可编程嵌入式片上系统，在单芯片中集成了 BLE 射频、可配置的模拟和数字外设功能、存储器以及双核 CPU 系统。由于集成了多项功能并支持各种低功耗模式，PSoC 6 BLE 是可穿戴设备电池供电、健康和运动 BLE 应用的理想选择。

8 相关应用笔记和代码示例

Table 3 列出了系统级和通用应用笔记，这些笔记建议用于下一步了解 PSoC 6 BLE 和 PSoC Creator。

Table 3.通用和 系统级应用笔记

文档编号	文档名称
AN218241	PSoC 6 MCU 硬件设计注意事项
AN219434	导入 PSoC Creator 代码到 PSoC 6 MCU 项目的 IDE
AN219528	PSoC 6 MCU 低功耗模式和功耗降低技术

Table 4 列出了器件的特定外设和应用的应用笔记和代码示例（CE）。

Table 4. PSoC 6 BLE 特性相关文档

文档	文档名称
蓝牙智能	
AN91162	创建 BLE 自定义配置文件
AN91445	天线设计和射频布局指南
AN92584	低功耗设计与 BLE 应用电池寿命估算
CE218463	BLE 告警通知客户端/服务器
CE218464	BLE 电话告警客户端/服务器
系统资源，CPU，和中断	
AN215656	PSoC 6 MCU 双核 CPU 系统设计
AN217666	PSoC 6 MCU 中断
CE216795	PSoC 6 MCU 双核基础
CE216825	PSoC 6 MCU 实时时钟基础
CE218129	PSoC 6 MCU 使用低功耗比较器从休眠中唤醒
CE218541	PSoC 6 MCU 故障处理基础
CE218542	PSoC 6 使用 RTC 警报中断的自定义节拍定时器
CE218552	PSoC 6 MCU 使用 DMA 的 UART 到 存储器缓存
CE218964	PSoC 6 MCU RTC 每日告警
CE219339	PSoC 6 MCU MCWDT 和 RTC 中断 (双核)
CE219521	PSoC 6 MCU GPIO 中断
CE219881	PSoC 6 MCU 切换功耗模式
CE220060	PSoC 6 MCU 看门狗定时器
CE220061	PSoC 6 MCU 多计数器看门狗中断
CE220120	PSoC 6 MCU 周期性中断
CE220169	PSoC 6 MCU 使用 TCPWM 的周期性中断
GPIO	

文档	文档名称
CE219490	PSoC 6 使用 SMART IO 呼吸 LED
CE219506	PSoC 6 使用 SMART IO 的时钟缓冲器
CE220263	PSoC 6 MCU GPIO 引脚示例
CapSense	
AN92239	CapSense 接近感应
Bootloader	
AN213924	MCU Bootloader 软件开发套件 (SDK) 指南
CE213903	PSoC 6 MCU 基本引导加载程序
通信	
CE220541	PSoC 6 MCU SCB EzI2C
音频	
CE218636	PSoC 6 MCU Inter-IC Sound (I2S) 示例
CE219431	PSoC 6 MCU PDM-to-PCM 项目
RTOS	
CE217911	PSoC 6 FreeRTOS™ 示例项目
安全	
CE220465	PSoC 6 MCU 加密技术 – AES 演示
CE220511	PSoC 6 MCU 加密技术 – SHA 演示

Appendix A. 赛普拉斯专业术语

本部分列出了在操作赛普拉斯 PSoC 器件系列时可能遇到的常用术语。

组件配置工具：是指每个组件中嵌入的 PSoC Creator 的简单图形用户界面（GUI）。通过该工具，您可以自定义组件参数，并且可以通过右键点击某个组件来访问它。

组件：由 PSoC Creator 软件中的一个图标表示的免费嵌入式芯片。它们用于将多个芯片和系统接口集成到已通过主系统总线与 MCU 连接的 PSoC 组件上。例如，通过 BLE 组件，可以在几分钟内构建 Bluetooth Smart 设备。同样，您可以将可编程模拟组件作为传感器使用。

MiniProg3：是指用于进行开发的硬件，即在不支持内置编程器的自定义电路板或 PSoC 开发套件上编程 PSoC 器件。

PSoC：PSoC 是一个包含 CPU（如 32 位 Arm Cortex-M0）、可编程的模拟和数字模块的可编程嵌入式设计平台。通过它，可以使用稳定且易于使用的解决方案（如触摸感应）来加快嵌入式系统的设计，并能够创建低功耗设计。

PSoC 6 BLE：它是一个集成了 BLE 射频的 PSoC 4 芯片。该 BLE 射频模块包括一个符合蓝牙 4.2 规范的买断式授权 BLE 协议栈。

PSoC Creator: 是 PSoC 3、PSoC 4， PSoC 5LP，和 PSoC 6 BLE 的集成开发环境（IDE）软件，该软件安装在电脑上，通过使用它可同时设计 PSoC 系统的硬件和固件，或允许设计硬件，然后将这些设计移植给其他流行 IDE 软件。

PSoC Programmer：是用于编程 PSoC 器件的灵活的集成式编程应用程序。PSoC Programmer 和 PSoC Creator 用于对 PSoC 3、PSoC 4、ProC， PSoC 5LP 和和设计进行编程。

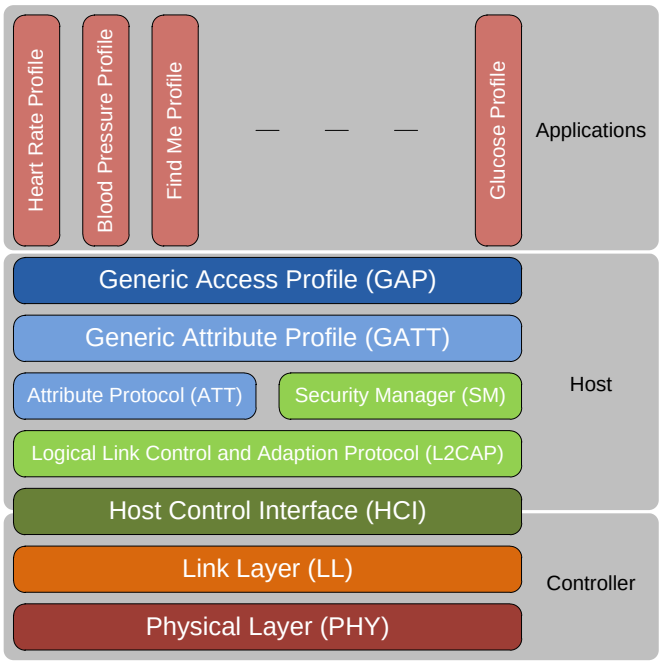
PSoC 6 BLE: 具有双核 CPU 和集成 BLE 无线的 PSoC 6 MCU，其中包含与蓝牙 4.2 规范兼容的免版税 BLE 协议栈

Appendix B. BLE 协议

B.1 概述

BLE，又称 Bluetooth Smart，是由蓝牙 SIG 引入的一个低功耗无线标准，其工作频带为 2.4 GHz ISM。 Figure 61 显示了 BLE 堆栈。

Figure 61. BLE 架构



BLE 堆栈可分为以下三部分：

- **控制器：**它是一个物理器件，用于编码数据包，并把该数据包作为无线信号进行传输。作为接收端使用时，控制器对无线信号进行解码，并重新构建数据包。
- **主机：**软件堆栈包括各种协议和配置文件（安全管理器、属性协议等），用于管理两个或多个器件间相互通信的方式。
- **应用：**指的是一个使用场合，其中通过使用软件堆栈和控制器执行特定功能。

下面各节是对 BLE 堆栈中各层的简介，并使用了标准的心率和电池服务作为示例。欲了解 BLE 架构描述的详细信息，请参考蓝牙开发者网页上的蓝牙 4.2 规范或观看相关的培训视频。

B.2 物理层 (PHY)

物理层使用了高斯频移键控（GFSK）调制方式在 2.4 GHz ISM 频带上发送并接收数字数据，数据速率为 1 Mbps。BLE 物理层将 ISM 频带分为 40 个 RF 通道（通道间隔为 2 MHz），其中有 37 个数据通道，剩下为 3 个广播通道。

B.3 链路层 (LL)

链路层通过执行关键流程来建立一个可靠的物理链路（通过使用确认和基于流控的架构），并用于增强 BLE 协议和降低功耗的性能。下面列出的是链路层的几项功能：

- 广播、扫描、创建和保持连接，用以建立物理链路
- 24 位 CRC 和 128 位 AES 加密，用于执行稳定安全的数据交换
- 为低功耗操作创建快速连接和低占空比广播
- 自适应跳频（AFH），它改变了用于数据包传输的通信通道，从而降低了其他器件的干扰

在链路层可定义两部分内容：

- **主设备**：以智能手机为例，它能够对主设备配置中的链路层进行配置。
- **从设备**：以心率监视设备为例，它能够对从设备配置中的链路层进行配置。

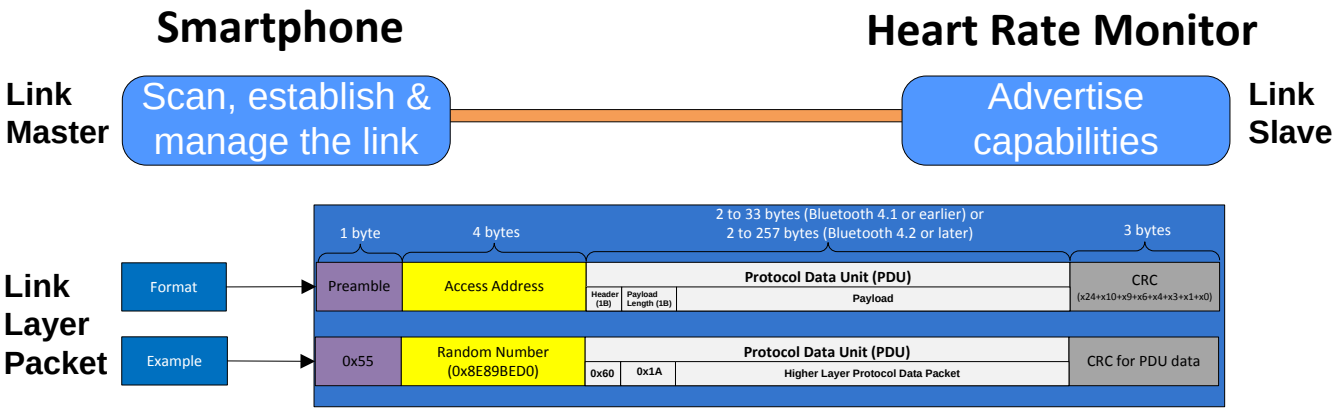
PSoC BLE/PRoC BLE 器件可以按照上述任意配置运行。

链路层从设备将它的存在信息广播给另一个链路层主设备。链路层主设备会接收广告数据包，然后根据具体应用的要求选择需要连接的从设备（请参见 Figure 62）。在心率监示器应用的示例中，心率监示器作为从设备使用，它将数据发送给作为主设备的智能手机。这时，智能手机应用将显示智能手机中数据的读取过程。

PSoC BLE/PRoC BLE 器件在硬件中执行链路层中对时间要求严格并且处理器密集型的任务，如广播、CRC 以及 AES 加密。链路层控制操作（如进入广播状态和启动加密）都是通过固件实现的。

Figure 62 显示的是 BLE 链路层数据包的结构以及链路层数据包中各字段的大小。链路层数据包将所有上层数据保存在其负载域内。它具有一个 4 字节的访问地址，可用于识别物理链路上进行的通信，并会忽略正在同一个 RF 通道中运行的相邻 BLE 器件的数据包。24 位 CRC 提供了稳定的数据。

Figure 62. BLE 链路层协议



B.4 主机控制接口 (HCI)

HCI 是主机与控制器间的标准定义接口。通过它，主机和控制器在进行不同的物理传输方式（如 USB 或 UART）中可以交换命令、数据和事件等信息。仅在该控制器和主机为不同的器件时，HCI 才需要进行物理传输。

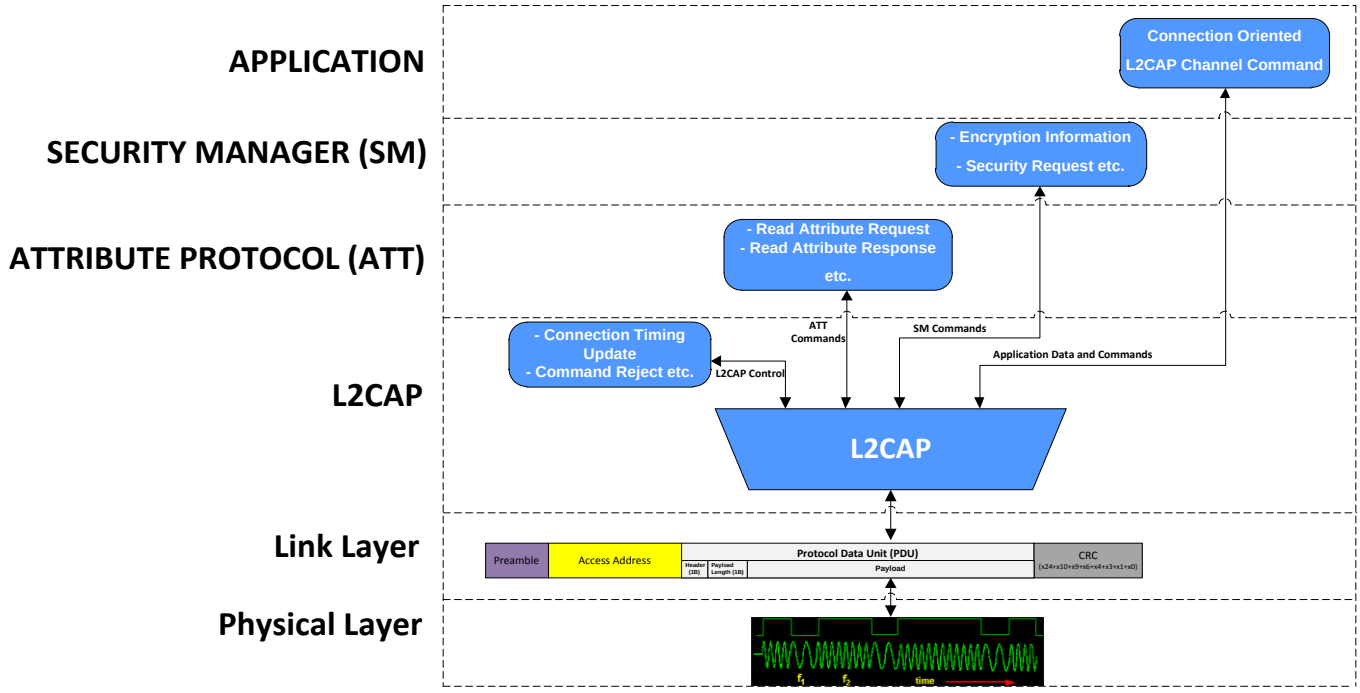
在 PSoC /PRoC BLE 器件中，HCI 作为一个固件协议层用于控制器和主机间信息和事件的传输。

B.5 逻辑链路控制及适配协议 (L2CAP)

L2CAP 为上层协议提供了协议复用、分段和重组服务。分段功能会将接收到的来自上层的数据包分为链路层可传输的更小的数据包，而重组功能会将接收到的来自链路层的小数据包结合成一个有意义的数据包。L2CAP 层为属性协议(ATT)、安全管理器(SM)和 L2CAP 控制支持三个协议通道 ID，如 Figure 63 所示。蓝牙 4.2 通过这些协议通道顶层上的 L2CAP 定向连接通道允许直的接数据通道。

L2CAP 及其上面各层在 PSoC /PRoC BLE 器件的固件内实现。

Figure 63. BLE L2CAP 层



B.6 安全管理器(SM)

SM 层定义了用于配对、加密和关键分配的方法。

- **配对**是使能安全性能的过程。在该过程中，会验证两个器件，对其链接进行加密，然后交换加密密钥。这样可以通过 BLE 接口安全交换数据而不会被 RF 通道上的无声监听器窃听。
- **绑定**是一个过程，其中密钥和识别信息在保存配对过程中被交换。器件绑定完毕后，重新连接这些器件时不需要再次进行配对过程。

BLE 使用了 128 位 AES 对数据进行加密。

B.7 属性协议(ATT)

为了了解 ATT 和 GATT 层，您应该掌握 BLE 中 GATT 的两种使用情况：

- **GATT 服务器：**是指包含数据或信息的设备。它接收 GATT 客户端设备的请求，并使用数据进行响应。例如，心率监示器的 GATT 服务器包含心率信息；BLE HID 键盘的 GATT 服务器包含用户按键信息。

- **GATT 客户端：**是指对 GATT 服务器发送请求和/或接收数据的设备。例如，智能手机是接收心率服务器心率信息的 GATT 客户端；手提电脑是接收 BLE 键盘按键信息的 GATT 客户端。

ATT 构成 BLE 通信基础。该协议使 GATT 客户端能够在 GATT 服务器中找到数据和属性并访问它。有关 GATT 客户端和服务器架构的更多信息，请参考通用属性配置文件(GATT)。

属性包含的是 ATT/GATT 内的基本数据，它包含了以下内容：

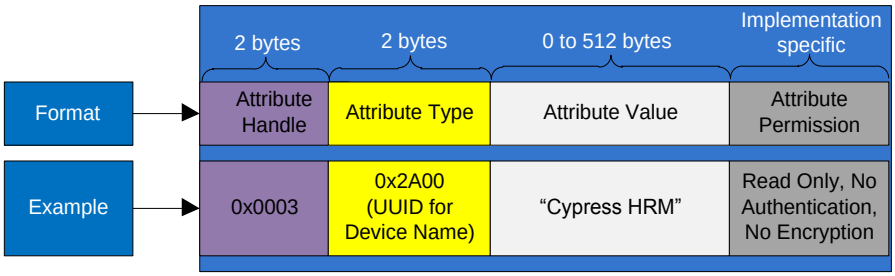
- **属性句柄：**这个 16 位的地址用于寻址和访问某项属性。
- **属性类型：**用于指定在某项属性中储存的数据类型。它由一个蓝牙 SIG 定义的 16 位 UUID 表示。

  例如，心率服务的 16 位 UUID 表示为 0x180D；器件名称属性的 UUID 表示为 0x2A00。请访问蓝牙网页，以获取 SIG 所分配的一系列 16 位 UUID。

- **属性值：**储存在属性中的实际数据。
- **属性许可：**用于指定属性访问、验证和授权等要求。属性许可由较高层规范设置，不能通过属性协议查看属性许可。

Figure 64 显示的是一个器件名称属性结构示例。

Figure 64. 属性格式示例



B.7.1 属性层次结构

属性是用于代表 ATT/GATT 中数据的构建模块。属性大致可分为以下两组，用于提供属性层次结构和数据提取：

- **特性：**显示系统信息或有意义的数据的属性集。一个特性包括以下各属性：
  - 特性声明属性：它定义了一个特性的开始。
  - 特性值属性：它保持了实际数据。
  - 特性描述符属性：它们是可选属性，提供了特性值的其他信息。

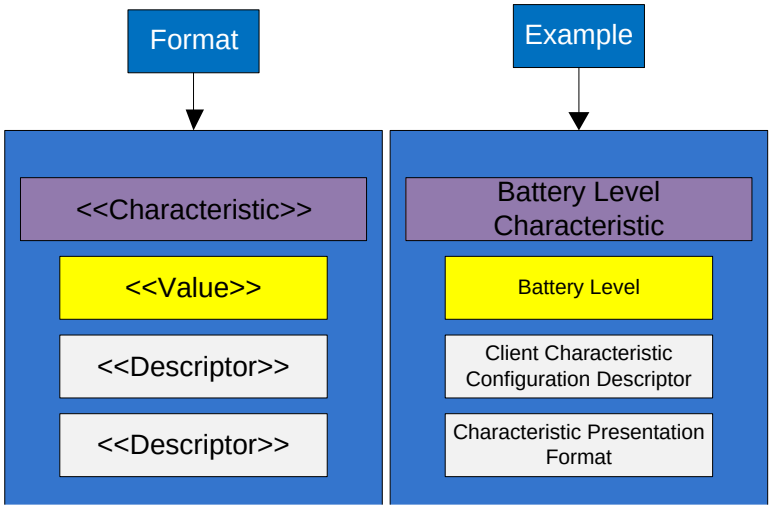
“电池电量”是电池服务（BAS）中的某个特性示例。以百分比值的形式显示电池电量便是一个特性描述符示例。

Figure 65 显示的是一个特性结构和电池电量特性示例。

- 某种特性的第一部分是特性声明（它表示特性的开始）内容，该声明由 Figure 65 中的电池电量指示。
- 后面是实际的特性值或实际数据，在电池电量特性示例中表示当前电池的电量。电池电量按全量程的百分比形式显示，如‘65’、‘90’等。
- 特性描述符提供了额外信息，用于描述特性值的意义。例如，电池电量的特性表示格式描述符指示：电池电量被表示为一个百分比值。因此，读取‘90’时，客户端会将其理解为 90%，而不是 90 mV 或 90 mAh。同样，有效范围特性描述符（不显示在 Figure 65 中）表示电池电量的范围为 0 至 100%。

- 通常，一个客户端特性配置描述符（CCCD）还可作为特性描述符使用，这样通过 GATT 客户端可以配置 GATT 服务器的特性行为。当 GATT 客户端将 0x01 值写入到某个特性的 CCCD 内时，便能够使 GATT 服务器发送异步通知（将在下一部分进行描述）。在电池电量特性中，将 0x01 值写入到电池电量 CCCD 内会使电池服务定期通知它的电池电量或电池电量值发生的所有变化。

Figure 65. 特性格式与示例

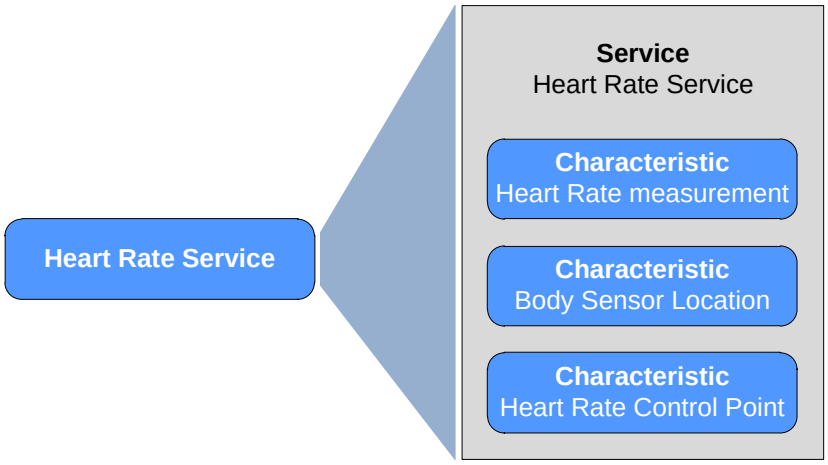


- **服务：**属性的类型，它定义了由 GATT 服务器所执行的功能。服务是一个特性集，它也可能包含其他服务。服务用于建立相关数据组并提供数据的层次结构。请参考 Figure 66，了解心率服务（HRS）示例。

  服务可分为两种类型：主要服务或辅助服务。主要服务显示了器件的主要功能，而辅助服务则提供了额外功能。例如，在心率监示器中，心率服务（HRS）是主要服务，电池服务（BAS）是辅助服务。

  一个服务还可以包含 GATT 服务器上存在的其他服务。所包含的全部服务组成新服务的一部分。

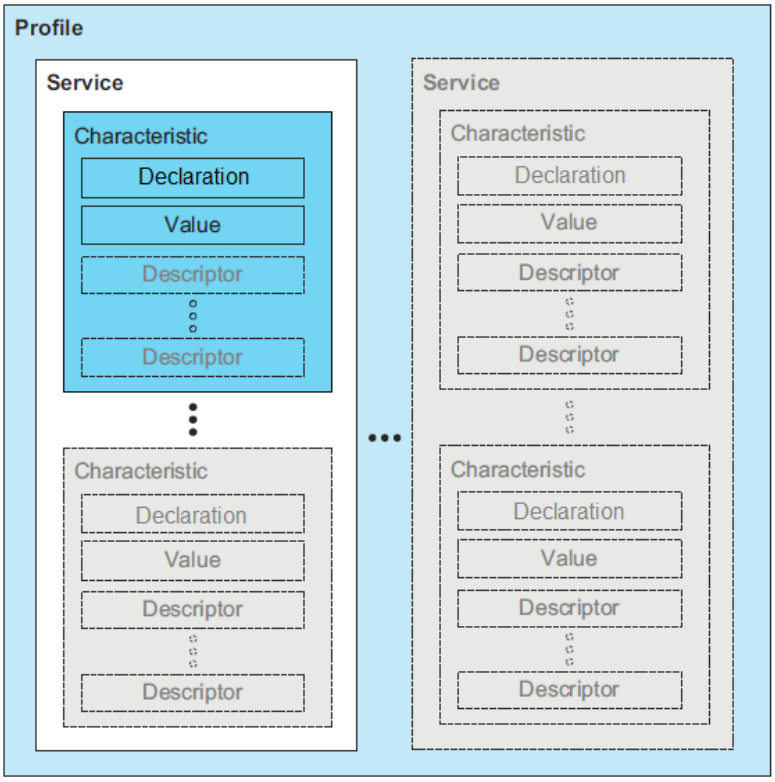
Figure 66. BLE 心率服务示例



BLE 中的“配置文件”指的是一个服务集和这些服务共同执行某个特定终端应用时的行为。心率配置文件（HRP）是 BLE 配置文件的一个示例，该 BLE 配置文件定义了创建心率监示器所需要的所有服务。更多详细信息，请参考通用访问配置文件 (GAP)章节。

Figure 67 显示的是数据的层次结构，该数据层次结构使用了该章节中所定义的属性、特性、服务和配置文件。

Figure 67. BLE 数据的层次结构*



*图片由蓝牙 SIG 提供

B.7.2 属性操作

- **读取请求：**GATT 客户端将该请求发送给 GATT 服务器，以读取属性值。对于所有请求，GATT 服务器会向 GATT 客户端发出一个响应。智能手机读取心率监示器的电池电量（请参考 Figure 65）便是读取请求的一个示例。
- **写入请求：**GATT 客户端向 GATT 服务器发送该请求，以写入一个属性值。GATT 服务器会向 GATT 客户端发出响应，以表示是否写入了该值。智能手机会将数值 0x01 写入到表示电池电量特性的 CCCD 内，以使能通知作为写入请求的一个示例。
- **写入命令：**GATT 客户端向 GATT 服务器发送该命令以写入一个属性值。对于该命令，GATT 服务器不会发送任何响应。例如，BLE 即时警报服务（IAS）使用一个写命令从一个 IAS 定位器（例如，一部智能手机）来触发 IAS 目标器件（例如，使用 BLE 的密钥卡）上的警报（打开 LED、使蜂鸣器报鸣、驱动振动电机等）。
- **通知：**GATT 服务器将其发送给 GATT 客户端，从而通知属性新值。GATT 客户端不会对一个通知发送任何响应。例如，将数值 0x01 写入到心率监示器的 CCCD 内时，该心率监示器会向智能手机发送心率测量通知。
- **指示：**GATT 服务器负责发送该类型的信息。GATT 客户端始终会确认该信息。例如，BLE 健康温度计服务（HTS）使用各指示可靠地向健康温度计收集器（如智能手机）发送所测量的温度值。
-

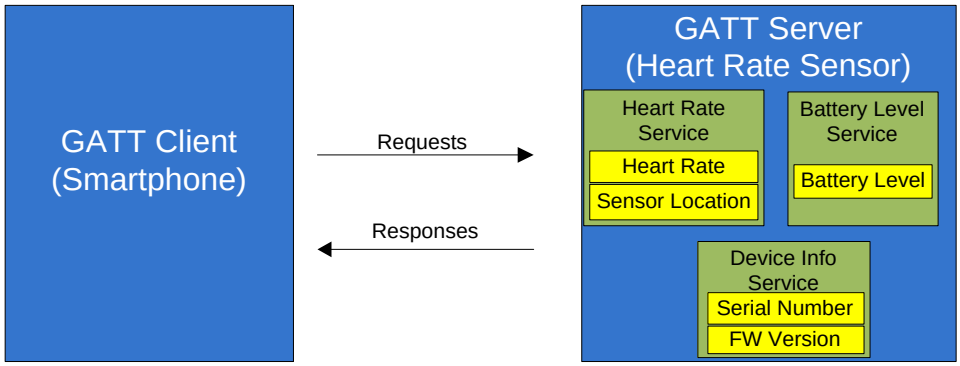
B.8 通用属性配置文件(GATT)

GATT 定义了查找和使用属性的方式。GATT 可以按下面两种方式中的一种进行操作：

- **GATT 客户端：**是指请求数据的器件（例如智能手机）
- **GATT 服务器：**是指提供数据的器件（例如心率监示器）

Figure 68 显示了 GATT 层上的客户端-服务器架构，它以心率监测设备为例。该心率监示器提供了多项服务（HRS、BAS 和器件信息服务）。每项服务都包含一个或多个带有特性值和描述符的特性，如 Figure 65.所示。

Figure 68. GATT 客户端-服务器架构



以链路层级别建立 BLE 连接后，GATT 客户端（最初并不了解已连接的 BLE 器件）会启动一个被称为“服务发现”的过程。作为服务发现过程的一部分，GATT 客户端会向 GATT 服务器发送多个请求，从而获取 GATT 服务器中包含了所有可用服务、特性以及属性的列表。服务发现过程完成后，GATT 客户端将具有所需信息，从而通过使用前一章节所介绍的属性操作来修改或读取 GATT 服务器所显示的信息。

B.9 通用访问配置文件 (GAP)

GAP 层提供了设备特定的信息，如设备地址、设备名称以及发现、连接、绑定的各种方法。配置文件定义了一个设备的发现和连接方法、可用服务的列表，以及服务的使用方式。Figure 70 显示的是一个心率配置文件示例。

GAP 可以按照下面四种方式中的一种进行操作：

- **外设：**它起着广播作用，使设备能够连接至 GAP 中心。建立与 GAP 中心的连接后，器件会作为从设备进行操作。例如，心率传感器会将所测量的心率报告给远程设备，使之作为一个 GAP 外设进行操作。
- **中心：**它作为 GAP 使用，用于扫描广播和启动同外设间的连接。建立与外设的连接后，该 GAP 作为主设备运行。例如，一个外设（心率传感器）检测心率测量数据的智能手机作为 GAP 中心操作。
- **广播员：**它作为一个发送器，用于发送数据该功能不能实现 BLE 连接和进行数据交换（无请求/响应操作）它的工作原理类似于一个广播电台：它会不断发送数据，却不需关心是否有人收听，它进行的是单向数据通信。作为一个信标是 GAP 广播员的典型示例，它会不断播放信息而无需任何响应。
- **观察员：**作为一个“听众”，它会扫描广播，但不会连接到广播的设备。观察员角色与广播员角色完全相反。它的工作原理类似于一个无线收音机，可以一直收听信息而不会向信息源发送任何响应。智能手机中可连续监听各种信标的应用便是一个 GAP 观察器的典型示例。

Figure 69 显示的是一个通用 BLE 系统，其中，赛普拉斯的 BLE Pioneer 套件作为外设，一台智能手机作为中心设备。此外，该图还说明了各 BLE 协议层间的交互以及它们在中心设备和外设中所起的作用。

Figure 69. BLE 系统设计

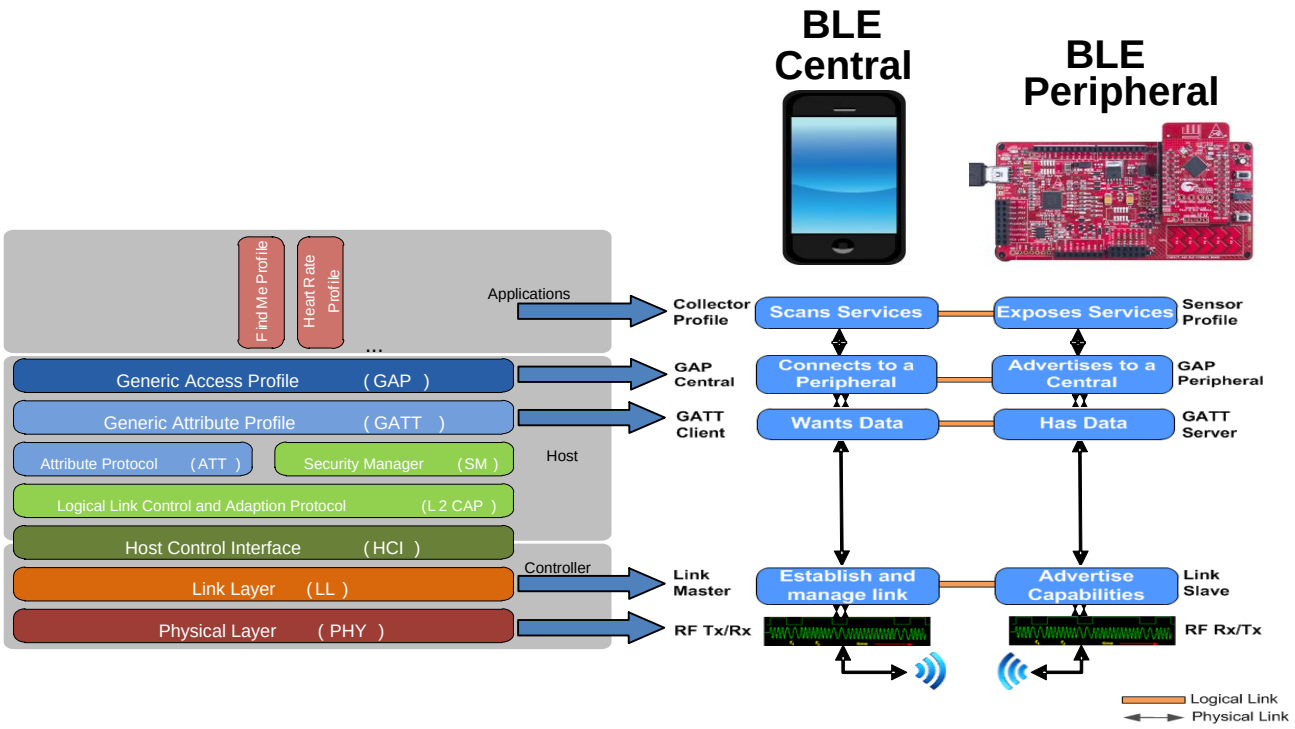
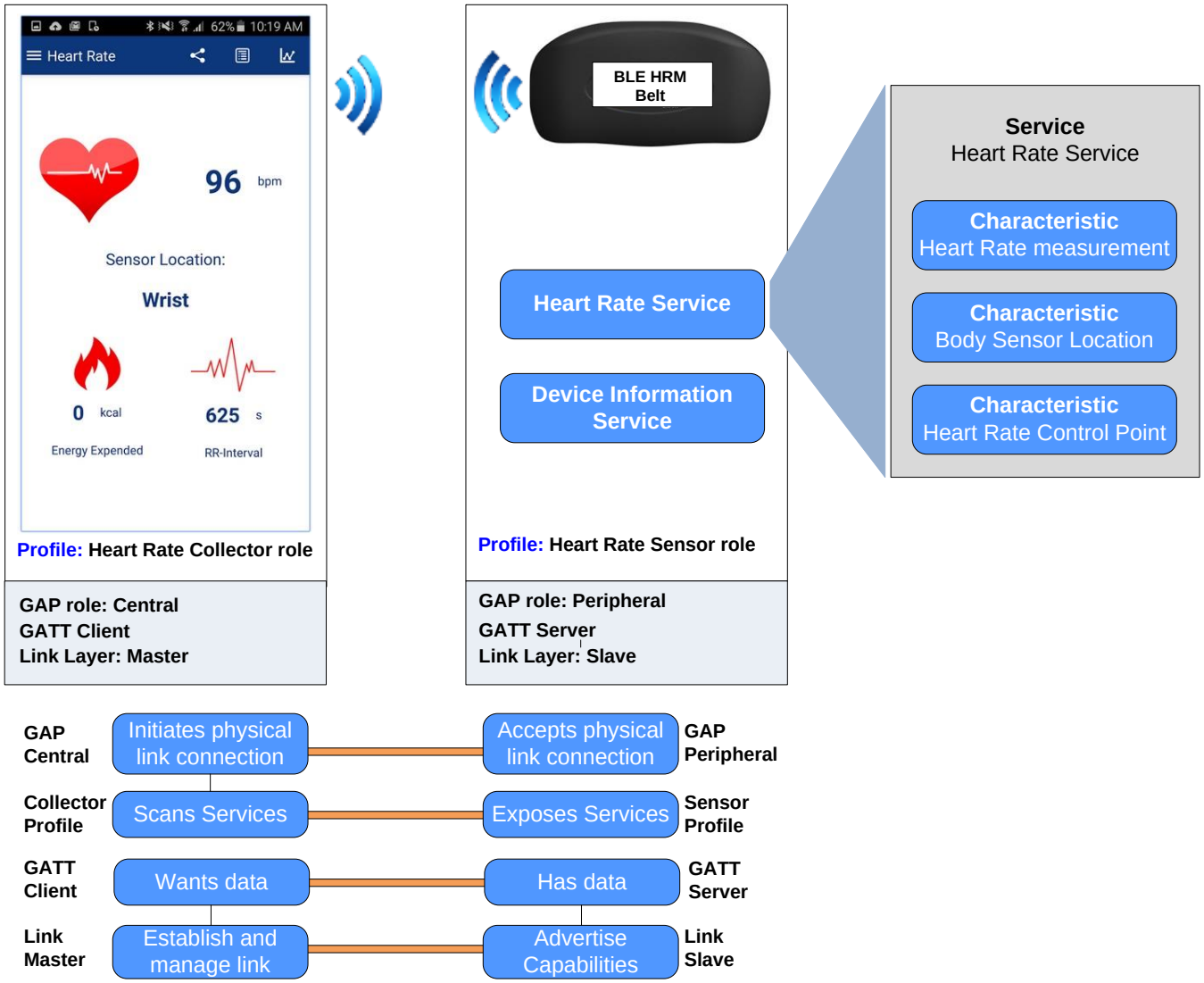


Figure 70 显示的是一个示例，其中，一个带有心率应用的智能手机作为中心设备使用，另外一个心率传感器作为外设使用。监示心率的设备将执行心率传感器的配置文件，而接收数据的智能手机会执行心率收集器的配置文件。

在该示例中，心率传感器配置文件实现了两项标准的服务。第一项是心率服务，它包含三项功能，分别为心率测量功能、身体感应位置功能和心率控制点功能。第二项是设备信息服务。在链路层上，心率测量器件作为从设备，智能手机作为主设备。有关心率服务和配置文件的详细消息，请参见蓝牙开发人员的入口网站。

Figure 70. BLE 心率监测系统



Appendix C. PSoC 6 MCU BLE 器件特性

C.1 系统资源

C.1.1 CPU 子系统: Arm Cortex-M4 和 Arm Cortex-M0

PSoC 6 BLE 中的 CPU 子系统由两个 Arm Cortex 内核组成：Cortex-M 4，带有一个能在最高频率 150 MHz 运行的单精度浮点单元; Cortex-M 0 +能够在最高频率为 100 MHz 运行。两个内核中都有存储器保护单元（MPU）可用。此外，附加到外设的保护单元叫做外设保护单元（PPU），对于共享存储器区域，提供共享存储器保护单元（SMPU）。

Cortex-M0+也提供安全无中断的启动功能。这允许在执行用户应用程序之前检查系统完整性并强制执行权限。

C.1.2 IPC

处理器间通信（IPC）为两个内核提供了通信和同步其活动的功能。 IPC 硬件使用 PSoC 6 BLE 中的寄存器结构实现。这些寄存器结构用于同步事件，并触发 IPC 通道的“通知”或“释放”事件。 PSoC 6 BLE 支持多达 16 个通道，允许在 CPU 内核之间传递消息并支持锁定以实现互斥。

C.1.3 存储器系统

CPU 内核具有固定的内存地址映射，可以共享内存和外设。 代码可以在两个内核的闪存和 RAM 上执行。

PSoC 6 BLE 系列具有高达 1 MB 的闪存和额外的可用于 EEPROM 仿真的 32 KB 闪存。 还有 32 KB 的安全闪存可以通过一次性编程的密钥锁定和访问。 此外，闪存支持 RWW（Read-While-Write）操作，以便在 CPU 正在执行指令时写入闪存。 还有一个 128 KB 的 ROM，包含启动和配置程序。 如果终端应用程序需要验证用户闪存，这将确保安全引导操作。

PSoC 6 BLE 具有高达 288 KB 的 SRAM 存储器，能够完全保留或以用户指定的 32 KB 模块增量保留。

C.1.4 DMA

CPU 子系统包括两个独立的 DMA 控制器，每个控制器都可以运行 32 个通道。 这些控制器支持使用 Arm 标准高级微控制器总线架构（AMBA）高性能总线（AHB）对外设进行独立访问。

DMA 将数据在内存、外设和寄存器之间传输。 这些传输独立于 CPU。 DMA 通道支持以下内容：

- 源和目标的 8 位、16 位和 32 位数据宽度
- 每个通道有四个优先级
- 每个 DMA 描述符的可配置中断
- 描述符链接

C.1.5 时钟系统

PSoC 6 BLE 提供以下时钟源：

- **内部主振荡器（IMO）：**在 PSoC 6 BLE 中，IMO 是主要的内部时钟源。CPU 和所有高速外设都由 IMO 或外部晶振（ECO）提供时钟。PSoC 6 BLE 的多个外设时钟分频器均由 IMO 或 ECO 提供时钟，这些时钟为高速外设生成时钟。IMO 可以生成频率为 8 MHz 准确度为±1%的时钟，只在活动模式下可用。
- **外部晶振器(ECO)**: PSoC 6 器件包含一个振荡器，用于驱动外部 4 MHz 至 33.33 MHz 晶振以获得精确的时钟。该时钟源使用 PSoC 中的振荡器电路构建。 该电路采用外部晶体，需要在 PSoC 器件的外部晶振引脚上进行填充。

BLE 子系统上的外部晶体振荡器具有内置的可调谐晶体负载电容，用于生成高精度的 32 MHz 时钟。它主要用于为产生 RF 时钟的 BLE 子系统提供时钟。高精度 ECO 时钟也可用作 PSoC 6 BLE 器件高频时钟（CLK_HF）的时钟源，并被指定为 AltHF 时钟。

- **外部时钟(EXTCLK)**: 外部时钟是一个兆赫范围的时钟，可以来自指定 I / O 引脚上的信号。该时钟可以用作 PLL 或 FLL 的源时钟，也可以由高频时钟直接使用。
- **内部低速振荡器（ILO）**：ILO 是一款超低功耗的 32-kHz 振荡器，主要为在所有功耗模式下工作的低速外设生成时钟。
- **精密内部低速振荡器（PILO）**：PILO 是一种额外的信号源，可以提供比 ILO 更准确的 32.768 kHz 时钟。 PILO 在深度睡眠模式和更高模式下工作。
- **时钟晶体振荡器（WCO）**：32.768 kHz 的 WCO 被用作 CLK_LF 的源（与 ILO / PILO 一起）。 WCO 用于精确维护 BLE 广告和连接事件的时间间隔。与 ILO 类似，WCO（除休眠和停止模式外）还可用于所有模式。

PSoC BLE 时钟生成系统具有可用于生成高频时钟（CLK_HF）的锁相环（PLL）和频率锁定环（FLL）模块。这些高频时钟依次驱动 CPU 内核时钟和外设时钟分频器。 PSoC 6 BLE 在 CLK_HF [0]和 WCO 上具有时钟监控器，用于检测时钟是否已停止并可触发中断或系统复位，或同时触发中断和系统复位。

PSoC 6 BLE 具有五个高频根时钟（CLK_HF [0-4]）。每个 CLK_HF 在器件上都有一个特定的目标，如串行存储器接口等外设。

C.1.6 系统中断

PSoC 6 BLE 在两个 CPU 内核上支持中断和异常：Cortex-M4 和 Cortex-M0 +。 内核提供自己的矢量表来处理中断/异常。 PSoC 6 BLE 可支持 Cortex-M4 上的多达 139 个中断以及 Cortex-M0 +上的 32 个中断。 多达 33 个中断可以将器件从深度睡眠模式唤醒。

C.1.7 电源和监控

PSoC 6 BLE 器件系列支持 1.71 V 至 3.6 V 的工作电压。它集成了单输入多输出（SIMO）降压转换器，为器件内的模块供电。内核工作电压可在 0.9 V 至 1.1 V 之间由用户选择。器件系列支持多个电源轨 - V_{DDD}，V_{DDA}，V_{DDIO} 和 V_{BACKUP}。此外，还有用于 BLE 无线电操作的电源轨 - V_{DDR}。应用中各种电源轨/引脚的可用性取决于所选的器件封装。

该器件包括一个 V_{BACKUP} 电源引脚，用于为 RTC 和 WCO 等一小组外设供电。该导轨独立于所有其他导轨，即使没有其他导轨时也可以存在。由于这些模块的电源来自专用的 V_{BACKUP} 引脚，因此即使断开设备电源或保持复位状态，这些模块仍可继续工作。备用域中的 RTC 提供了一个选项，可以将设备从任何功耗模式唤醒。

PSoC 6 BLE 系列支持上电复位（POR）、掉电检测（BOD）、过压保护（OVP）和电源监控的低压检测（LVD）电路，并实现故障安全恢复。

有关 PSoC 6 BLE 中电源的更多信息，请参考 PSoC 6 MCU: PSoC 63 BLE 架构技术参考手册。

C.1.8 超低功耗模式

按照功耗降序，PSoC 6 BLE 支持的功耗模式为：

- **活动模式**：这是主要的操作模式。在此模式下，所有外设都处于活动状态并可用
- **低功耗活动模式**：该模式类似于活动模式，大多数外设以有限的功能运行; CPU 核心可用。性能权衡包括降低工作时钟频率，限制高频时钟源以及降低内核工作电压。
- **睡眠模式**：在此模式下，CPU 内核处于睡眠模式，SRAM 处于保留状态，并且所有外设均可用。任何中断唤醒任一 CPU 并将系统返回到活动模式。 Cortex-M4 和 Cortex-M0 +均支持各自的 CPU 睡眠模式，每个 CPU 可独立于另一 CPU 的状态进入睡眠状态。当两个内核都处于 CPU 睡眠状态时，该器件被认为处于睡眠模式。
- **低功耗睡眠模式**：大多数外设以有限的能力运行; CPU 内核不可用。
- **深度睡眠模式**：在深度睡眠模式下，所有高速时钟源均关闭。这反过来使得高速外设在深度睡眠中不可用。只能在低频时钟上工作的外设可用。
- **休眠模式**：器件和 GPIO 状态被冻结，器件在唤醒时复位。

您可以在电池供电的 BLE 系统中结合使用睡眠模式，深度睡眠模式和休眠模式，以获得业界最佳的系统电源和更长的电池寿命。 Table 5 显示了 PSoC 6 BLE 系统功耗模式与 BLE 子系统（BLESS）功耗模式之间的依赖关系。 Table 5 单元格中的复选标记表示当系统处于给定功率模式时，BLESS 可以执行指定的任务。 例如，当系统处于低功耗活动模式时，BLESS 可以处于深度睡眠模式。 保留标签表示当系统切换到深度睡眠模式时，BLESS 操作上下文保留。

Table 5. PSoC 6 BLE 功耗模式

BLESS 模式	PSoC 6 BLE 系统功耗模式					
	活动	低功耗活动	睡眠	低功耗睡眠	深度睡眠	休眠
发送	✓	✓	✓	✓	保留	关闭
接受	✓	✓	✓	✓	保留	关闭
空闲	✓	✓	✓	✓	保留	关闭
深度睡眠	✓	✓	✓	✓	保留	关闭

C.2 安全引导

安全引导涉及使用特定市场/应用程序的标准定义的基于密钥的安全协议来验证应用程序闪存映像。 安全引导过程由安全标准管理，并在 PSoC 6 BLE 器件的独立监控闪存中实施。 引导代码计算安全闪存引导代码的 AES-CMAC 并将其与 eFuse 中的值进行比较。 如果这些值不匹配，则启动过程失败。 引导代码还会执行 eFuse 中指定的调试访问限制。

C.3 可编程数字外设

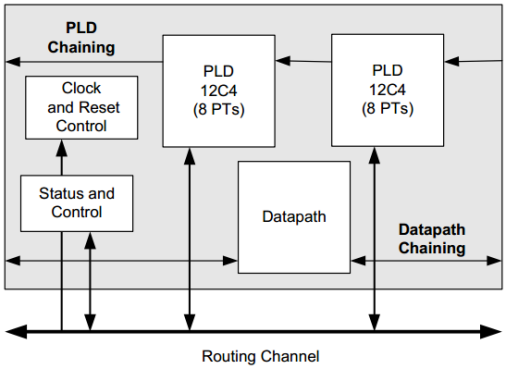
C.3.1 UDB

UDB 是可编程逻辑模块，提供类似于 CPLD 和 FPGA 模块的功能，如 Figure 71 所示。 UDB 允许您创建各种数字功能，如定时器，计数器，PWM，伪随机序列（PRS），CRC，移位寄存器，SPI，UART，I^2S 以及定制组合和时序逻辑电路。

每个 UDB 有两个可编程逻辑器件（PLD），每个具有 12 个输入和 8 个产品条件。 PLD 可以形成注册或组合的积之和逻辑。 此外，每个 UDB 中都有一个 8 位单周期算术逻辑单元（ALU），称为“数据通路”。 数据通路有助于高效实现定时器，计数器，PWM 和 CRC 等功能。

UDB 还提供交换式数字信号互连（DSI）结构，允许将来自外设和端口的信号路由到 UDB 并通过 UDB 进行通信和控制。

Figure 71. 通用数字框图



您不一定需要知道任何硬件描述语言（HDL）才能使用 UDB。 赛普拉斯的 PSoC 6 BLE 开发工具 PSoC Creator 可以从原理图为您生成所需的功能。 如果需要，高级用户可以使用 Verilog 在 UDB 上实现自定义逻辑。

PSoC 6 BLE 最多有 12 个 UDB。 有关 UDB 的更多信息，请参阅以下应用笔记。

- AN62510 –使用 PSoC 3，PSoC 4 和 PSoC 5LP 实现状态机
- AN82156 – PSoC 3，PSoC 4 和 PSoC 5LP - 使用 UDB 数据路径设计 PSoC Creator 组件

- AN82250 – PSoC 3, PSoC 4, 和 PSoC 5LP – 用 Verilog 实现可编程逻辑设计

C.3.2 可编程 TCPWM

PSoC 6 BLE 具有 32 个可编程 TCPWM 模块。 每个 TCPWM 可以实现一个定时器、计数器、PWM 或正交解码器功能。 TCPWM 提供死区可编程互补 PWM 输出和可选的启动、重新加载、停止、计数和捕捉事件信号。 PWM 模式支持中心对齐，边沿和伪随机操作。 它也有一个 Kill 输入来强制输出到预定状态。

有关更多信息，请参考 PSoC 6 BLE TCPWM 组件数据手册。

C.3.3 SCB

PSoC 6 BLE 具有多达 9 个具有 I2C，SPI 或 UART 的独立运行时可编程 SCB。 SCB 支持以下功能：

- 采用 Motorola®，Texas Instruments® 安全简单配对（SSP）和 National Semiconductor® Microwire 协议的标准 SPI 主机和从机功能
- 带有智能卡读卡器，单线本地互联网络（LIN）接口，智能卡（ISO7816）和红外数据协会（IrDA）协议的标准 UART 功能（高达 1 Mbps 波特率）
- 标准 I2C 主机和从机功能，工作速度高达 1 Mbps
- EzSPI 和 EzI2C 模式，无需 CPU 干预即可运行
- 一个 SCB 可以配置为使用外部时钟在深度休眠模式下工作。仅在从模式下，SPI 和 I2C 协议（使用外部时钟）支持低功耗（深度睡眠）操作模式。

有关更多信息，请参阅 PSoC 6 BLE SCB 组件数据手册。

Table 6 l 列出了数字外设的一些 PSoC BLE 应用。

Table 6.可编程数字外设应用

应用	数字外设
传感器集线器	I²C（数字传感器接口）、PWM（执行器）、I2S（语音输入）
健康与健身	PWM（用户界面）、计数器（波形峰值测量）、SPI（外部存储器接口）
家庭自动化	PWM（车库门控制）、PWM（灯光控制）

C.3.4 BLESS

PSoC 6 BLE 集成了一个蓝牙智能子系统，可实现蓝牙 4.2 规范中规定的 BLE 链路层和物理层。 BLE 子系统包含具有嵌入式 AES-128 安全引擎的物理层（PHY）和链路层引擎。该子系统支持所有 Bluetooth SIG 采用的 BLE 配置文件。

物理层由符合蓝牙 4.2 规范的数字 PHY 和 RF 收发器组成。收发器通过 2.4-GHz ISM 频段以 1 Mbps 发送和接收 GFSK 数据包。基带控制器是一种支持主模式和从模式的复合硬件/固件实现。 HCI 和链路控制等关键协议元素以固件形式实现，而时间关键功能（如加密，CRC，数据白化和访问码关联）则以硬件实现。

BLESS 符合蓝牙 4.2 规范，支持蓝牙 4.0 规范的所有功能以及蓝牙 4.2 规范的一些附加功能，如低占空比广告，LE ping，L2CAP 面向连接的信道，链路层隐私，链路层数据长度扩展和 LE 安全连接。 BLESS 模块还包含 ECO 和 WCO，它们分别用于生成精确的 RF 频率并保持 BLE 链路上连续的连接间隔之间的时间。

BLE 子系统最多支持四个同时连接。它支持自己的四种功能模式：深度睡眠，空闲，发送和接收。

注意：本节中讨论的功率模式针对 BLESS 模块。对于 PSoC 6 BLE 系统功耗模式，请参见超低功耗模式。

C.3.4.1 深度睡眠模式

深度睡眠模式是 BLESS 支持的最低功耗功能模式。在此模式下，无线关闭。此模式用于在数据包发送和接收完成后的广告或连接间隔期间最大限度节省功耗。在此模式下可以关闭 ECO 以节省电力; WCO 是低频时钟，用于维护 BLE 链路层定时参考逻辑。应用程序固件控制进入和退出该状态。

C.3.4.2 睡眠模式

在睡眠模式下，无线关闭。该模块保留所有配置。 ECO 和 WCO 已打开，但核心 BLESS 逻辑的时钟已关闭。应用程序固件控制进入和退出该状态。

C.3.4.3 空闲模式

空闲模式是发送和接收状态的准备状态。在此状态下，无线电被关闭，但链路层时钟为链路层逻辑启用，以便 CPU 启动协议状态机。

C.3.4.4 发送模式

发送模式是主动功能模式; BLESS 内的所有模块都已打开。使能链路层时钟以完成链路层和 RF-PHY 内的逻辑。在这种模式下，RF-PHY 从链路层获得高达 2 Mbps 的串行数据，并将 2.4 GHz GFSK 调制数据传输到天线端口。 BLESS 从空闲模式进入发送模式。

C.3.4.5 接收模式

该模式使 BLESS 进入接收状态，执行特定于 BLE 的接收器操作。 RF PHY 转换从 RF 模拟模块接收的 1 Mbps 数据，并在解调后将其转发给链路层控制器。Table 7 给出了 BLESS 功率模式和运行子模块的总结。

Table 7. BLESS 功耗模式

BLESS 功耗模式	ECO	WCO	RF Tx	RF Rx	BLESS Core
深度睡眠	关	开	关	关	关
睡眠	开	开	关	关	关
空闲	开	开	关	关	开
发送	开	开	开	关	开
接收	开	开	关	开	开

C.3.5 音频子系统

PSoC 6 BLE 具有一个由 I2S 模块和两个 PDM 通道组成的音频子系统。 PDM 通道连接到数字麦克风的比特流输出。PDM 处理通道提供下垂校正，并且可以在 384 kHz 至 3.072 MHz 范围内的时钟速度下工作，并以高达 48 ksps 的音频采样率产生 16 至 24 位字长。

当传输 8 位至 32 位字时，I2S 接口支持主从模式，字时钟速率最高可达 192 ksps。

C.3.6 串行存储器接口

PSoC 6 BLE 上的串行存储器接口（SMIF）能够连接不同类型的最多四个存储器。 SMIF 支持八进制 SPI（8 位/周期吞吐量），双四进制 SPI（8 位/周期吞吐量），Quad-SPI（4 位/周期吞吐量），DSPI（2 位/周期吞吐量）和 SPI（1 位/循环吞吐量）。该模块还支持就地执行（XIP）模式，其中 CPU 内核可以直接从外部存储器执行代码。 SMIF 模块以及在 PSoC Creator 中开发的软件模块使您能够在应用程序中使用合格的预定存储器设备。

C.3.7 eFUSE

eFuse 是一次性可编程的，用于编程安全相关的设置。它也可以用来存储一次编程和稍后使用的应用程序设置。

C.3.8 段式 LCD

PSoC 6 BLE 段式 LCD 驱动器具有以下功能：

- 支持多达 8 个通用（COM）和 64 段（SEG）电极
- 可编程 GPIO 提供 COM 和 SEG 电极的灵活选择
- 支持 14 段和 16 段字母数字显示，7 段数字显示，点阵和特殊符号
- 两种驱动模式：数字关联和 PWM
- 在除休眠以外的所有系统电源模式下运行

- 可以从 1.8 伏 VDD 驱动 3 伏显示器
- 数字对比度控制

注：PSoC 6 BLE 器件支持的公共和段数取决于器件系列和器件封装。有关详细信息，请参阅相应的器件数据手册。

C.4 可编程模拟外设

C.4.1 连续时间模块运算

PSoC 6 BLE 器件具有一对基于连续时间模块（CTBm）的运算放大器，其输入和输出连接到固定位置引脚。运算放大器支持除了休眠模式外的所有功耗模式。但是，在深度睡眠模式下，运算放大器的运行降低了增益带宽积。这些典型使用中的运算放大器的输出可用作 SAR 输入的缓冲器。

C.4.2 低功耗比较器

PSoC 6 BLE 器件具有一对低功耗比较器，它们也可以在深度睡眠和休眠模式下工作。在低功耗模式下，比较器的电流消耗小于 300 nA。在功耗敏感的设计中，当器件进入低功耗模式时，您可以使用低功耗比较器来监控模拟输入并产生可唤醒系统的中断。

有关更多信息，请参考 PSoC 6 BLE 低功耗比较器组件数据手册。

C.4.3 SAR ADC

PSoC 6 BLE 具有 12 位 1 Msps 逐次逼近型寄存器（SAR）ADC，输入通道支持可编程分辨率和单端或差分输入选项。GPIO 的数量限制了可以实现的 ADC 输入通道的数量。由于需要高速时钟，因此 SAR ADC 不能在深度睡眠和休眠模式下工作。

SAR ADC 有一个硬件序列器，可以在没有 CPU 干预的情况下在多达 8 个通道上执行自动扫描。 SAR ADC 可以通过 8 输入序列器连接到一组固定的引脚。序列器自主循环选择的通道（序列扫描），并以零切换开销进行操作。它还支持预处理操作，如在这八个通道上对输出数据进行累加和平均。

您可以使用各种方法触发扫描，如固件、定时器、引脚或 UDB，为您提供额外的设计灵活性。

为了在噪声条件下提高性能，可以在内部参考放大器的固定位置引脚上提供外部旁路。有关 SAR ADC 的更多信息，请参考 PSoC 6 BLE SAR ADC 组件数据手册。

PSoC 6 BLE 具有片上温度传感器，可用于测量温度。温度传感器可以连接到 SAR ADC，它通过赛普拉斯提供的包含校准和线性化的软件组件将读数数字化并产生温度值。

C.4.4 DAC

PSoC 6 BLE 具有 12 位电压模式连续时间 DAC（CTDAC），其建立时间为 2 µ s。 12 位 DAC 提供连续时间输出，无需外部采样保持（S / H）电路。 DAC 控制接口提供了一个通过 CPU 和 DMA 控制 DAC 输出的选项。这包括一个双缓冲 DAC 电压控制寄存器，用于可编程更新速率的时钟输入，DAC 缓冲区清空中断至 CPU，以及触发至 DMA。 DAC 可能因此由 DMA 控制器驱动以生成用户定义的波形。

有关 DAC 的更多信息，请参考 PSoC 6 BLE DAC 组件数据手册。

C.4.5 CapSense

PSoC 6 BLE 器件中的第四代 CapSense 支持基于自电容和互电容的触摸感应。 CapSense 通过可连接到模拟多路复用总线的 CSD 支持所有引脚。

电容式触摸传感器使用人体电容来检测传感器上或附近的手指是否存在。电容式传感器美观大方，易于使用，使用寿命长。 PSoC 6 BLE 中的 CapSense 功能提供了前所未有的 SNR; 最佳液体耐受性;以及各种传感器类型，如按钮、滑块、跟踪板和接近传感器。

赛普拉斯提供的软件组件使电容式传感设计变得非常简单;该组件支持称为 SmartSense 的自动硬件调整功能，并为触控板和接近传感器提供手势识别库。

有关更多信息，请参见 PSoC CapSense 设计指南。

CapSense 模块具有两个 7 位 IDAC，如果 CapSense 未使用该 IDAC，则可用于一般用途。 （慢）10 位斜率 ADC 可以通过使用其中一个 IDAC 来实现。有关如何使用斜率 ADC 的更多信息，请参见 PSoC 6 BLE CapSense 组件数据手册。

C.5 可编程 GPIO

I / O 系统在 CPU 内核、外设和外部器件之间提供接口。 PSoC 6 BLE 具有多达 104 个可编程 GPIO 引脚。您可以将 GPIO 配置为 CapSense、LCD、模拟或数字信号。 PSoC 6 BLE GPIO 支持多种驱动模式，驱动强度和转换速率。

PSoC 6 BLE 提供智能路由系统，可为内部信号连接至 GPIO 提供多种选择。这种灵活的布线简化了电路设计和电路板布局。

此外，PSoC 6 BLE 最多包含两个智能 I / O 端口，可用于对发往和来自 GPIO 引脚的信号执行布尔运算。

Appendix D. 赛普拉斯物联网开发工具

D.1 带 BLE Pioneer 先锋套件的 PSoC 63

Figure 72 所示的 PSoC 6 BLE Pioneer 套件是赛普拉斯 BLE 开发工具包，它支持 PSoC 6 BLE 系列器件。

以下是 PSoC 6 BLE Pioneer 套件基板的功能：

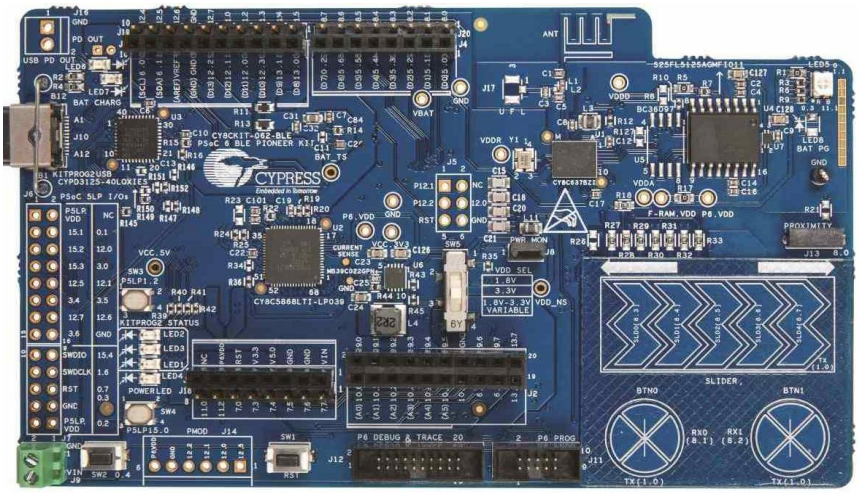
- 可以使用纽扣电池或通过 Type-C USB 接口供电。 Type-C USB 接口还支持高达 12 V，3 A 充电（PD）用户和供应商配置文件。
- 支持开发电池供电的低功耗 BLE 设计，这些设计可与标准的 Arduino Uno 连接器兼容的屏蔽或板上 PSoC 6 BLE 器件功能（例如 CapSense 用户界面和串行存储器接口）配合使用。
- 支持使用 Cortex Debug / ETM 连接器的第三方编程，调试和跟踪。
- 包含一个额外的标头，支持与来自第三方供应商（如 Digilent）的 Pmod™子卡连接。
- 支持 PDM-PCM 麦克风，实现 BLE 语音功能。
- 包括 QSPI NOR 闪存和 F-RAM。

该套件包括以下内容：

- 作为 BLE 链接主站的 USB-BLE 软件狗，与 CySmart 主机仿真工具配合使用，在非 BLE Windows PC 上提供 BLE 主机仿真平台
- 电子墨水显示屏

该套件由一套 BLE 示例项目和文档组成，可帮助您开始开发自己的 BLE 应用程序。访问 PSoC 63 with BLE Connectivity Pioneer Kit 网页以取得该套件的最新更新并下载该套件的设计、示例项目和文档文件。

Figure 72.带 BLE 连接先锋套件的 CY8CKIT-062-BLE PSoC 63



D.2 CySmart 主机仿真工具

该 CySmart 主机仿真工具是一个 Windows 应用程序，模拟使用 PSoC 6 BLE 先锋套件加密狗的 BLE 中心设备; 请参见 Figure 72。它作为 BLE Pioneer 套件安装的一部分进行安装，并可以通过右键单击 BLE 组件来启动。它提供了一个平台，允许您发现和配置外设上的 BLE 服务、特性、和属性，以测试 PSoC 6 BLE 外设通过 GATT 或 L2CAP 面向连接的通道实施的能力。

您可以使用 CySmart 主机仿真工具执行的操作包括但不限于：

- 扫描 BLE 外围设备以发现可连接的可用设备。
- 发现可用的 BLE 属性，包括连接的外设上的服务和特性。
- 对特征值和描述符执行读写操作。
- 从连接的外设接收特征通知和指示。
- 使用 BLE 安全管理器程序与连接的外围设备建立连接。
- 按照 Bluetooth 4.2 规范与外设建立 BLE L2CAP 面向连接的会话并交换数据。
- 对赛普拉斯 BLE 外设进行无线（OTA）固件升级。

Figure 73 和 Figure 74 显示了 CySmart 主机仿真工具的用户界面。有关如何设置和使用此工具的更多信息，请参阅帮助菜单中的 CySmart 用户指南。

Figure 73. CySmart 主机仿真工具主设备选项卡

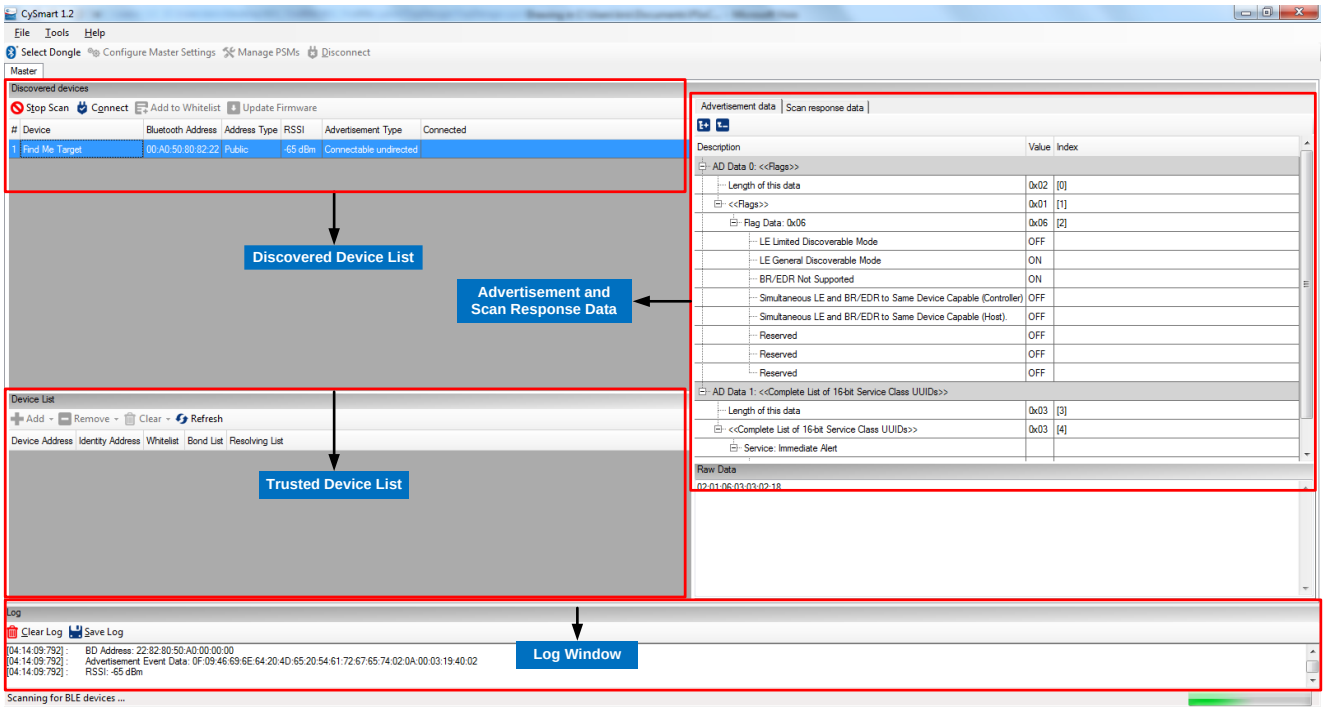
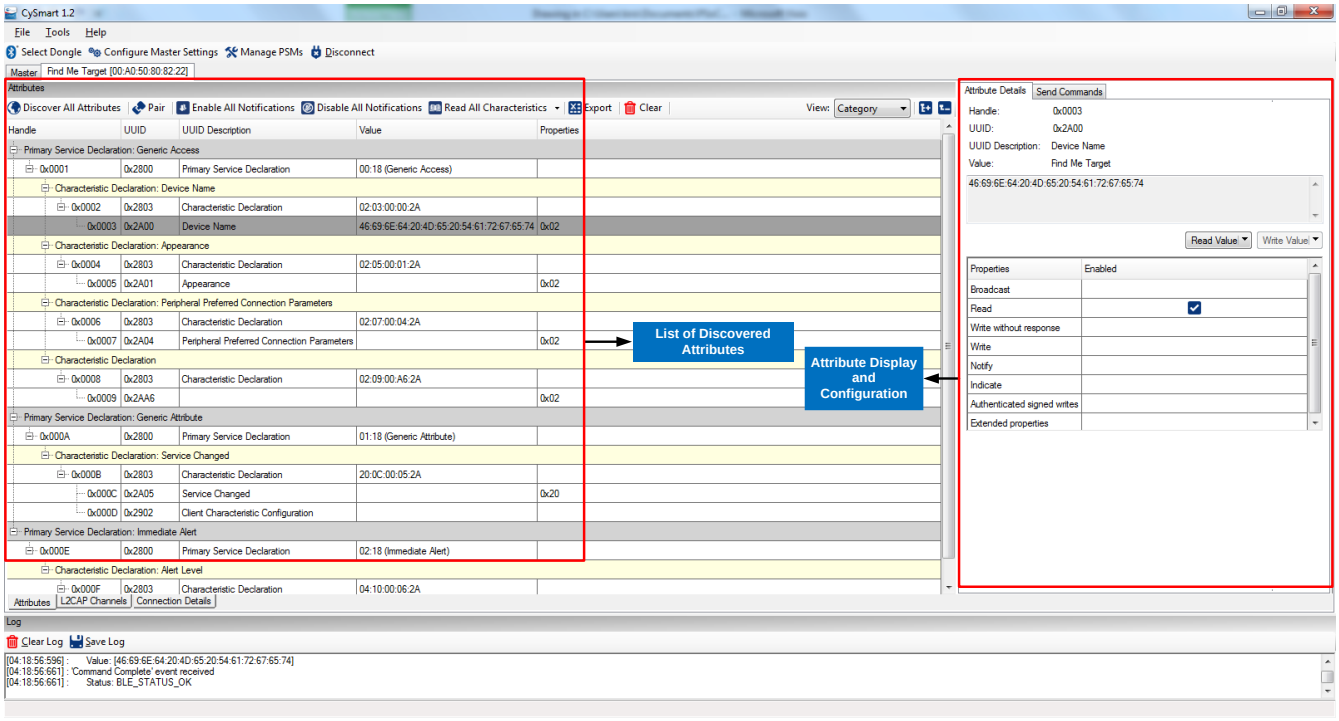


Figure 74. CySmart 主机仿真工具外设属性选项卡



D.3 CySmart 移动应用程序

除 PC 工具外，您还可以从各个应用商店下载适用于 iOS 或 Android 的 CySmart 移动应用。此应用程序分别使用 iOS Core Bluetooth 框架和 BLE Android 内置平台框架。它将启用 BLE 的智能手机配置为可扫描并连接到外设的中央设备。

该移动应用程序通过直观的 GUI 支持 SIG 采用的 BLE 标准配置文件，并简化了潜在的 BLE 服务和特性细节。除了 BLE 标准配置文件之外，该应用还演示了使用赛普拉斯 LED 和 CapSense 演示示例的定制配置文件实施。Figure 75 和 Figure 76 显示了心率配置文件用户界面的一组 CySmart 应用程序屏幕截图。有关如何在 BLE Pioneer 套件示例项目中使用该应用的说明，请参阅 BLE Pioneer 套件指南。

Figure 75. CySmart iOS App 心率配置文件示例

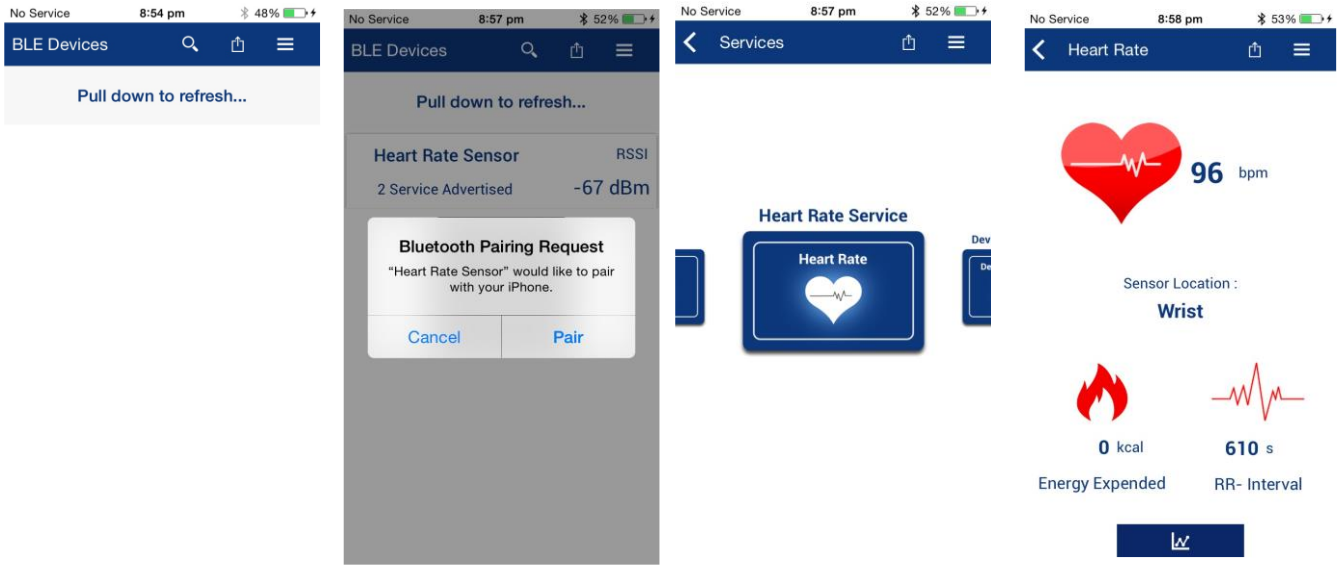
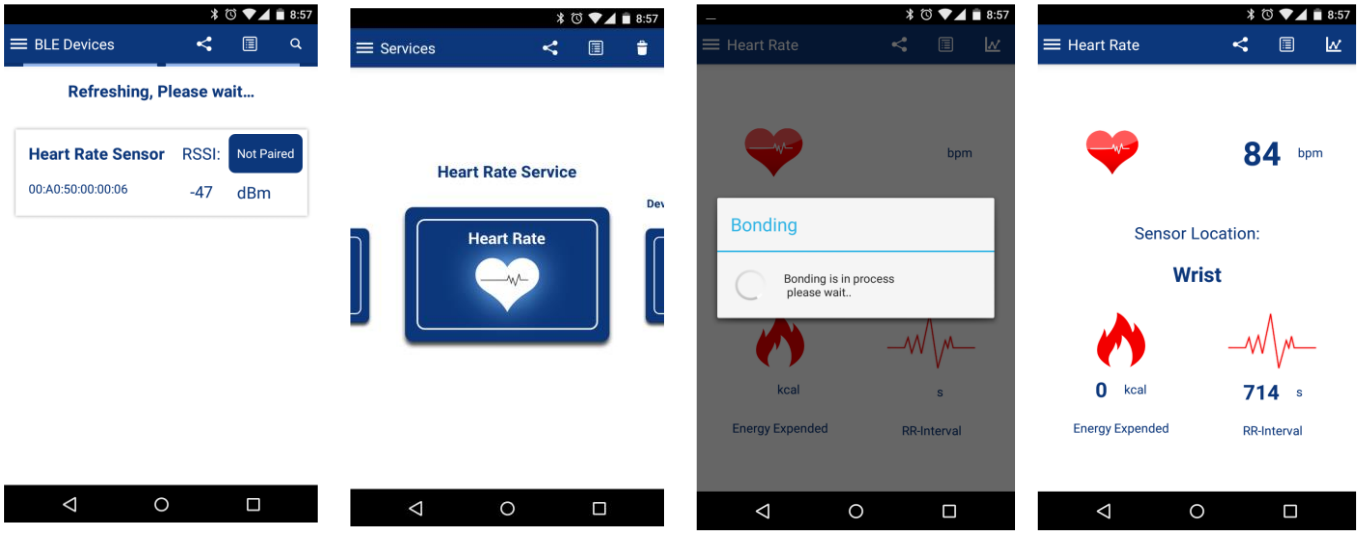


Figure 76. CySmart Android App 心率配置文件示例



修订记录

文档编号: AN210781 – 带有蓝牙低功耗（BLE）连接的 PSoC 6 MCU 入门

文档编号: 002-24354

版本	ECN	变更者	提交日期	变更说明
**	6223150	XITO	07/02/2018	本文档版本号为 Rev. **，翻译自 002-10781 Rev.B

销售、解决方案以及法律信息

全球销售和设计支持

赛普拉斯公司具有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。要想查找离您最近的办事处，请访问赛普拉斯所在地。

产品

Arm® Cortex® 微控制器	cypress.com/arm
汽车级产品	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
物联网	cypress.com/iot
存储器	cypress.com/memory
微控制器	cypress.com/mcu
PSoC	cypress.com/psoc
电源管理 IC	cypress.com/pmic
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线连接	cypress.com/wireless

PSoC®解决方案

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

赛普拉斯开发者社区

社区 | 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

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

© 赛普拉斯半导体公司，2016-2018 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。没有任何电子设备是绝对安全的。因此，尽管赛普拉斯在其硬件和软件产品中采取了必要的安全措施，但是赛普拉斯并不承担任何由于使用赛普拉斯产品而引起的安全问题及安全漏洞的责任，例如未经授权的访问或使用赛普拉斯产品。此外，本材料中所介绍的赛普拉斯产品有可能存在设计缺陷或设计错误，从而导致产品的性能与公布的规 格不一致。（如果发现此类问题，赛普拉斯会提供勘误表）赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。