PSoC™ Creator 上 PSoC™ 6 MCU 入门

相关器件系列

所有 **PSoC™ 6 MCU** 器件。

软件版本

PSoC™ Creator 4.2

关于本文档

范围及目的

AN221774 介绍 PSoC™ 6 MCU，这是一款采用 Arm® Cortex® M4 和 Cortex®-M0+处理器的双 CPU 可编程片上系统。本应用笔记可帮助您探索 PSoC™ 6 MCU 架构和开发工具，并向您展示如何使用 PSoC™ Creator 创建第一个项目。本应用笔记还指导您获取更多在线资源，以加快您对 PSoC™ 6 MCU 的学习。要学习使用带有 Bluetooth® Low Energy (LE)连接器件系列的 PSoC™ 6 MCU，请参考 **AN210781** – Getting Started with PSoC™ 6 MCU with Bluetooth® LE Connectivity on PSoC™ Creator。

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。请访问 **GitHub** 站点，获取使用 ModusToolbox IDE for PSoC™ 6 的全面代码示例集。您也可以在这里浏览 PSoC™视频库。

目录

相关器件系列 1
软件版本 1
关于本文档 1
目录 1
1 简介 3
1.1 先决条件 4
1.1.1 硬件 4
1.1.2 软件 4
2 开发生态系统 5
2.1 PSoC™资源 5
2.2 固件/应用开发 5
2.2.1 选择 IDE 6
2.2.2 PSoC™ Creator 6
2.2.2.1 PSoC™ Creator 帮助 7
2.2.3 PSoC™ 6 器件的软件开发套件 7
2.3 支持其他 IDE 8
2.4 RTOS 支持 8
2.4.1 使用 PSoC™ Creator 支持 RTOS 8
2.5 编程/调试 9
2.6 PSoC™ 6 MCU 开发套件 9

目录

3 器件特性 10
4 我的首个使用 PSoC™ Creator 的 PSoC™ 6 MCU 设计 12
4.1 使用这些说明 12
4.2 关于设计 12
4.3 第 1 部分: 从零开始创建新项目 13
4.4 第 2 部分: 实施设计 17
4.5 第 3 部分: 生成源代码 25
4.6 第 4 部分: 编写固件 26
4.7 第 5 部分: 构建项目并对设备进行编程 30
4.8 第 6 部分: 测试您的设计 32
5 总结 35
相关应用笔记和代码示例 36
术语表 38
文档修订记录 39

1 简介

PSoC™ 6 MCU 是超低功耗 PSoC™器件，具有针对智能家居，物联网网关等量身定制的双 CPU 架构。PSoC™ 6 MCU 器件是一款可编程嵌入式片上系统，集成了以下功能到单个芯片:

- 单 CPU 微控制器: Arm® Cortex®-M4 (CM4) 或双 CPU 微控制器: Arm® Cortex®-M4 (CM4) 和 Cortex®-M0+ (CM0+)。
- 可编程模拟和数字外设。
- 最多 2 MB 闪存和 1 MB SRAM。
- 第四代 CAPSENSE™技术。
- PSoC™ 6 MCU 适用于各种功耗敏感的应用，例如:
  - 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。

可编程模拟和数字子系统使用基于原理图的设计工具 **PSoC™** Creator，实现设计的灵活性和动态微调。

Figure 1 显示了使用 PSoC™ 6 MCU 的实际用例的应用级框图。

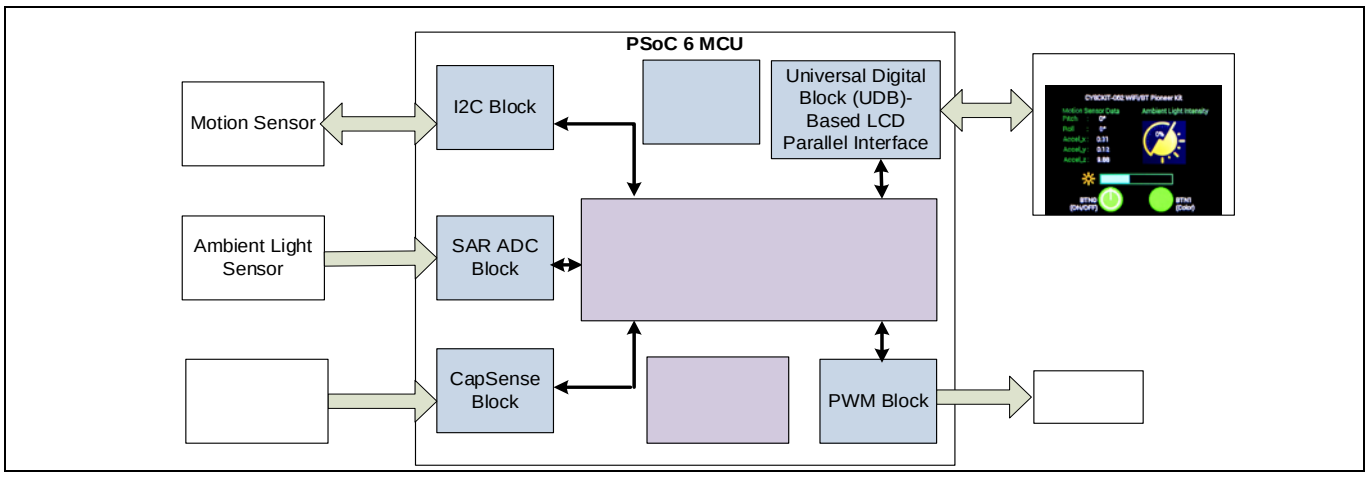

Figure 1 使用 PSoC™ 6 MCU 的应用级框图

PSoC™ 6 MCU 是一种功能强大且灵活的解决方案。例如，**Figure 1** 中的实际用例利用了以下功能:

- 降压转换器，用于超低功耗操作。
- 设备内的模拟前端 (AFE)，用于调节和测量传感器输出，如环境光传感器。
- 串行通信模块 (SCB)，用于连接多个数字传感器，如运动传感器。
- CAPSENSE™技术可实现可靠的触摸和接近感应。
- 数字逻辑 (通用数字模块或 UDB) 和外设 (定时器计数器 PWM 或 TCPWM) 分别驱动显示器和 LED。
- 由 CM0+ CPU 管理的产品安全功能和 CM4 CPU 执行的应用程序功能。

有关更多详细信息，请参见**器件功能**和器件数据表。

简介

本应用笔记向您介绍了 PSoC™ 6 MCU 的功能，概述了开发生态系统，并让您开始学习使用 PSoC™ 6 MCU 的简单设计。此设计可用作 PSoC™ Creator 的代码示例 **CE221773**。

有关硬件设计的注意事项，请参见 **AN218241 - PSoC™ 6 MCU 硬件设计注意事项**。

1.1 先决条件

在开始之前，请确保您已安装开发套件并已安装所需的软件。建议您下载代码示例以供参考。

1.1.1 硬件

- **CY8CKIT-062-WiFi-Bluetooth® PSoC™ 6 Wi-Fi-BT Pioneer Kit**
- **CY8CKIT-062-BLE PSoC™ 6 Bluetooth® LE Pioneer Kit** 或
- **CY8CPROTO-062-4343W PSoC™ 6 Wi-Fi BLUETOOTH Prototyping Kit** (PSoC™ Creator 不支持它)。

1.1.2 软件

- **PSoC™ Creator 4.2**，配置**外设驱动库** (PDL v3.1.x 或更高版本)
- **CE221773** –PSoC™ 6 MCU Hello World 示例

2 开发生态系统

2.1 PSoC™资源

www.infineon.com 上提供了大量数据，可帮助您选择合适的 PSoC™器件，并快速有效地将其集成到您的设计中。有关 PSoC™ 6 MCU 资源的完整列表，请参见 **How to Design with PSoC 6 MCU - KBA223067**。以下是 PSoC™ 6 MCU 的简要资源列表。

- 概述: **PSoC™ Portfolio**, **PSoC™ Roadmap**
- 产品选择器: **PSoC™ 6 MCU**
- **数据手册**: 描述并提供每个器件系列的电气规范。
- **应用笔记**和**代码示例**涵盖了从基础到高级的广泛主题。您还可以浏览我们的代码示例集合。请参阅**代码示例**。
- **技术参考手册 (TRM)** 提供了每个器件系列的架构和寄存器的详细说明。
- **PSoC™ 6 MCU 编程规范**提供了对 PSoC™ 6 MCU 器件的非易失性存储器进行编程所需的信息。
- **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-WiFi-BT PSoC™ 6 Wi-Fi-BT Pioneer Kit** 套件是一款开发套件，支持带 Wi-Fi 和 Bluetooth® 连接的 PSoC™ 62 系列 MCU。
  - **CY8CKIT-062-BLE PSoC™ 6 Bluetooth® LE Pioneer Kit** 是一款易于使用且价格低廉的开发平台，适用于具有 Bluetooth® LE 连接功能的 PSoC™ 63 系列 MCU。
  - **CY8CPROTO-062-4343W PSoC™ 6 Wi-Fi Bluetooth® Prototyping Kit** 是一款开发套件，支持 PSoC™ 62 系列 MCU 以及基于 CYW4343W 模块的 Wi-Fi 和 Bluetooth®连接，可在 ModusToolbox 上进行开发。
- 培训视频: 有关我们产品和工具的**视频培训**，包括 **PSoC™ 6 MCU** 专用系列。

2.2 固件/应用开发

英飞凌提供两个开发平台，您可以使用它们进行 PSoC™ 6 MCU 应用开发:

- **ModusToolbox:** ModusToolbox 软件包括配置工具，底层驱动程序，中间件库和操作系统支持，以及使您能够创建 MCU 和无线应用程序的其他软件包。还包括可选的 ModusToolbox IDE。

  ModusToolbox IDE 是基于 Eclipse 的开发环境，可在 Windows，macOS 和 Linux 平台上运行，并包含各种工具。

  ModusToolbox 支持完全集成到 IDE 中的独立设备和中间件配置器。使用配置器设置设备中不同模块的配置，并生成可用于固件开发的代码。 ModusToolbox 支持所有 PSoC™ 6 MCU 器件。建议您将 ModusToolbox 用于 PSoC™ 6 MCU 的所有应用程序开发。有关更多信息，请参见 **ModusToolbox Software Overview**。

  英飞凌在 **GitHub** 网站上提供了库和支持软件。某些资源所有开发人员都会使用。其它的资源将由特定生态系统中的开发人员使用。

  GitHub 上可用的英飞凌软件资源支持一个或多个目标生态系统：

  - MCU 和蓝牙 SoC 生态系统 – 用于 PSoC™ 6、Wi-Fi、蓝牙和蓝牙低功耗应用开发的全功能平台
  - Mbed OS 生态系统–提供嵌入式操作系统，传输安全性和云服务以创建连接的嵌入式解决方案
  - Amazon FreeRTOS 生态系统–使用软件库扩展了 FreeRTOS 内核，可轻松将小型低功耗设备安全地连接到 AWS 云服务

  ModusToolbox 工具和资源也可以在命令行中使用。有关详细文档，请参见 **Running ModusToolbox from the Command Line**（从命令行运行 ModusToolbox）。

开发生态系统

有关更多信息，请参见 **AN228571 – Getting Started with PSoC™ 6 MCU on ModusToolbox**（AN228571 – ModusToolbox 上的 PSoC™ 6 MCU 入门）。

- **PSoC™ Creator:** 英飞凌专有的 IDE，仅在 Windows 上运行。它支持一部分 PSoC™ 6 MCU 设备以及其他 PSoC™设备系列，例如 PSoC™ 3、PSoC™ 4 和 PSoC™ 5LP。

2.2.1 选择 IDE

ModusToolbox 是最新一代的工具集，包含 ModusToolbox IDE。 IDE 是基于 Eclipse 的，因此支持 Windows，Linux 和 MacOS 平台。该工具支持所有 PSoC™ 6 MCU 器件。相关的硬件和中间件配置程序也可以在所有三个主机操作系统上工作。

ModusToolbox IDE 当前不支持 PSoC™ 6 MCU 的某些功能，例如 UDB 和 USB 主机。英飞凌将来会发布新版本的 ModusToolbox，以支持这些功能并改善用户体验。

如果您具有基于 Eclipse 的工具的使用经验，并且想利用基于 Eclipse 的 IDE 的强大功能和可扩展性，或者希望在 Linux 或 macOS 上使用开发环境，请选择 ModusToolbox。如果要使用英飞凌 IoT 设备构建 IoT 应用程序，或者正在使用 PSoC™ Creator 不支持的 PSoC™ 6 MCU 设备，应当选择 ModusToolbox。

PSoC™ Creator 是使用已久的英飞凌专有工具，仅在 Windows 上运行。这个成熟的 IDE 包括一个图形编辑器，在组件的帮助下支持基于原理图的设计输入。 PSoC™ Creator 支持所有 PSoC™ 3、PSoC™ 4、PSoC™ 5LP 器件和一部分 PSoC™ 6 MCU 器件。这部分 PSoC™ 6 MCU 设备包括高达 1 MB 闪存的设备。

如果您倾向于使用图形编辑器进行设计输入和代码生成，并且 IDE 支持您计划使用的 PSoC™ MCU，或者如果您打算在 PSoC™ MCU 上使用 UDB，请选择 PSoC™ Creator。

2.2.2 PSoC™ Creator

PSoC™ Creator 是一个免费的基于 Windows 的集成设计环境 (IDE)。它汇集了多个数字、模拟和系统组件和固件来构建应用程序，并使您能够同时设计硬件和固件系统。使用 PSoC™ Creator，您可以在原理图上选择、放置和配置组件;编写 C /汇编源代码;并编程和调试设备。

如 **Figure 2** 所示，使用 PSoC™ Creator，您可以:

1. 从 **File**> **Code Example** ...菜单中浏览代码示例集合。
   a) 根据设备系列过滤示例。
   b) 从基于 **Filter by** 选项提供的示例菜单中进行选择。
   c) 使用下载按钮下载代码示例。
   d) 根据选择创建新项目。
2. 探索包含 100 多个组件的库。
3. 拖放组件以在主设计工作区中构建硬件系统设计。
4. 查看组件数据表。
5. 使用配置工具配置组件。
6. 使用 PSoC™硬件共同设计应用程序固件。

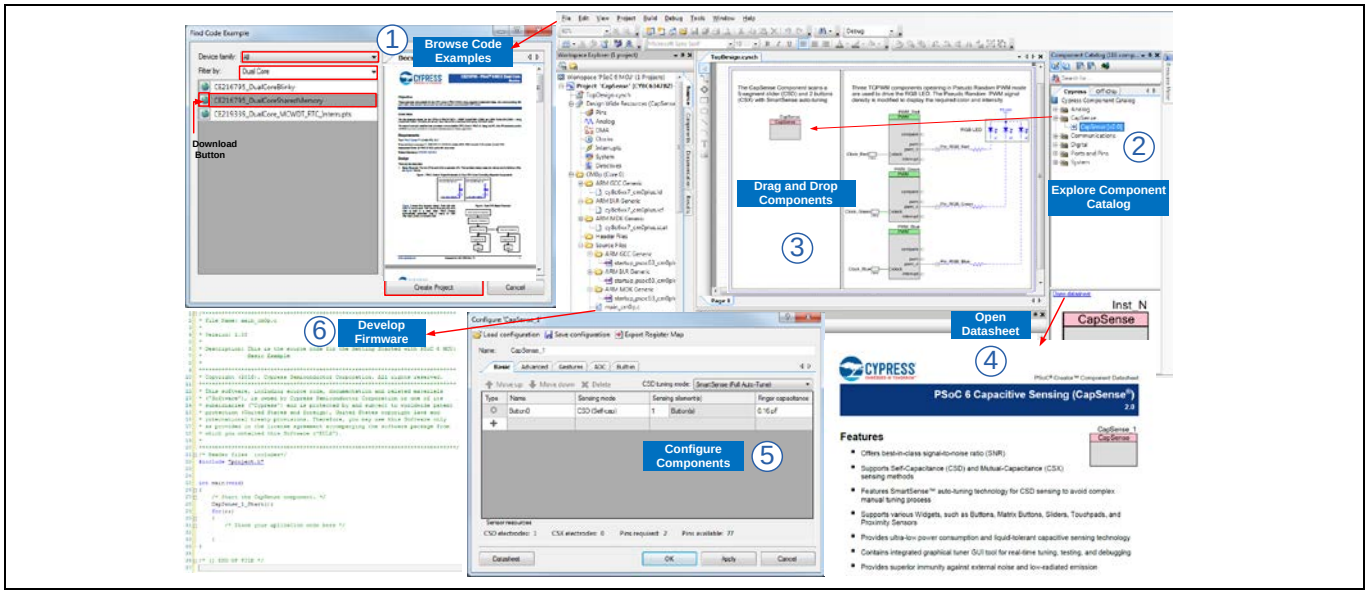

Figure 2 PSoC™ Creator 原理图条目和组件

2.2.2.1 PSoC™ Creator 帮助

访问 PSoC™ Creato 主页以下载并安装最新版本的 PSoC™ Creator。启动 PSoC™ Creator 并导航到以下项目:

- **快速入门指南**: 选择 **Help** > **Documentation** > **Quick Start Guide**。本指南为您提供了开发 PSoC™ Creator 项目的基础知识。
- **代码示例**: 选择 **File** > **Code Example** 或单击“**Start Page**”选项卡上的“**Find Code Example...**”链接。这些代码示例演示了如何配置和使用 PSoC™资源。
- **组件数据表**: 右键单击组件，然后选择打开数据表。访问 **PSoC™ 6 MCU 组件数据表**页面，获取所有组件数据表的列表。

2.2.3 PSoC™ 6 器件的软件开发套件

英飞凌提供重要的源代码和工具，以支持 PSoC™ 6 MCU 的软件开发。您可以使用工具指定您希望如何配置硬件，为固件使用的目的生成代码，以及包含各种中间件库以获得其他功能，如 Bluetooth® LE 连接或 FreeRTOS。此源代码可以更轻松地为受支持的设备开发固件。它可以帮助您快速定制和构建固件，而无需了解寄存器集。

对于 PSoC™ Creator 环境，英飞凌提供外设驱动程序库 (PDL)。PDL 支持 PSoC™ Creator 和第三方 IDE。您可以使用 PSoC™ Creator 组件来配置硬件。PSoC™ Creator 根据您的选择生成配置代码。该代码基于 PDL 驱动程序中的源代码。PDL 还包括各种中间件库。可能有 (也可能没有) 组件来帮助配置该代码。

驱动程序代码作为 *psoc6pdl* 库提供。中间件以 *psoc6mw* 的形式提供。无论是与 PSoC™ Creator 还是 ModusToolbox IDE 一起提供，PDL 源代码基本相同。这两个 IDE 存在实现差异。

中间件的提供方式有所不同。例如，PSoC™ Creator 中的 CAPSENSE™功能作为组件提供。对于 ModusToolbox 软件，有一个 Configurator 和一个中间件库。有关相同内容和不同内容的详细信息，请参阅两个 IDE 的相应文档。

无论您使用 PSoC™ Creator 还是第三方 IDE，希望在寄存器级别工作的固件开发人员还应使用 PDL 中的驱动程序源代码。PDL 包含项目所需的所有特定于设备的头文件和启动代码。它还可作为每个驱动程序的参考。由于 PDL 是作为源代码提供的，因此您可以看到它如何在寄存器级访问硬件。

某些设备不支持特定外设。PDL 是任何受支持设备的所有驱动程序的超集。这个超集设计意味着:

开发生态系统

- 用以初始化、配置和使用外设所需的所有 API 元素可用。
- 无论可用的外设如何，PDL 都适用于各种 PSoC™ 6 MCU 器件。
- PDL 包括错误检查，以确保所选设备上存在目标外设。

只要外设可用，这使代码能够保持 PSoC™ 6 器件系列的某些成员之间的兼容性。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受支持的外设，则在编译时会出现错误。在编写代码以使用外设之前，请查阅特定器件的数据手册以确认对外设的支持。

PSoC™ Creator 提供基于 PDL 的组件。这保留了 PSoC™ Creator 在利用英飞凌或社区开发和预验证组件方面的精髓。但是，PDL 是可以与任何开发环境一起使用的源代码库。

PDL 包括以下关键软件资源:

- 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。
- 特定于设备的头文件、启动和配置文件。
- 支持的第三方 IDE 的模板项目。
- 完整文档，可在*<PDL install directory>\doc*中找到。

有两个关键文件:

“PDL v3.x 用户指南”介绍了使用 PDL 的基本原理，如下所示:

- 使用 PDL (包括第三方 IDE) 创建自定义项目。
- 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。

PDL 3.x API 参考手册。此参考提供有关 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。

2.3 支持其他 IDE

您还可以使用您喜欢的 IDE (如 IAR Embedded Workbench) 为 PSoC™ 6 MCU 开发固件。英飞凌建议您使用配置工具生成资源配置。对于 PSoC™ Creator，配置是 IDE 的组成部分。

PSoC™ Creator 用于设置和配置 PSoC™ 6 MCU 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。如果设备配置发生更改，则在 PSoC™ Creator 中编辑 TopDesign 原理图并重新生成目标 IDE 的代码。

您可以在大多数 (如果不是全部) IDE 中有效地工作。如果 Target IDEs 面板不支持您的 IDE，您仍然可以使用 PSoC™ Creator。生成代码后，将必要的文件直接添加到 IDE 的项目中。**AN219434 - PSoC™ 6 MCU 将生成的代码导入 IDE**，提供了将生成的代码手动导入另一个 IDE 的详细步骤。

2.4 RTOS 支持

2.4.1 使用 PSoC™ Creator 支持 RTOS

PDL 包括对 PSoC™ 6 MCU 开发的 RTOS 支持: FreeRTOS 源代码完全集成并包含在 PDL 中。您可以使用 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，从 **Peripheral Driver Library** > **FreeRTOS** 下的 **Software package imports** 选项中选择 **FreeRTOS**，如 **Figure 3** 所示。

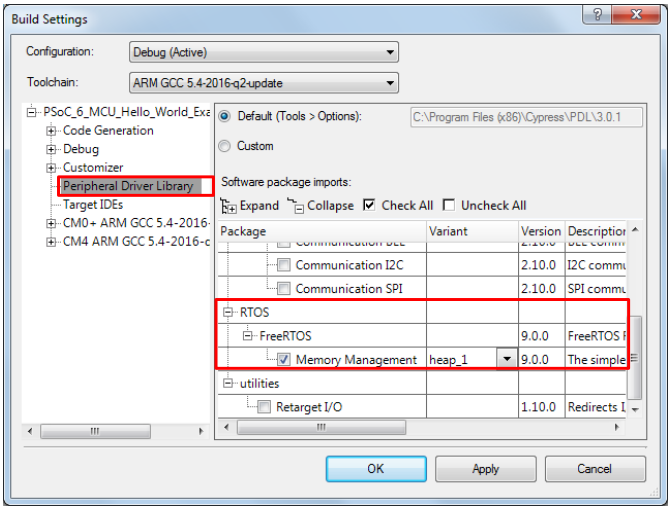

Figure 3　　导入 FreeRTOS 到 PSoC™ Creator 项目中

如果您有首选 RTOS，请使用提供的资源作为如何将此类代码与 PDL 集成的示例。

2.5　　编程/调试

PSoC™ 6 Wi-Fi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT) 和 **PSoC™ 6 Bluetooth® LE Pioneer Kit (CY8CKIT-062-BLE)** 具有 KitProg2 板载编程器/调试器。它支持 Cortex®微控制器软件接口标准 - 调试访问端口 (CMSIS-DAP) 和自定义操作模式，以及 KitProg2 连接。这使得调试 PSoC™ 6 MCU Pioneer 套件非常灵活。有关详细信息，请参阅“**KitProg2 用户指南**”。

PSoC™ 6 Wi-Fi Bluetooth®原型开发套件具有 KitProg3 板载编程器/调试器（CY8CPROTO-062-4343W）。它支持 Cortex®微控制器软件接口标准 - 调试访问端口 (CMSIS-DAP)。有关详细信息，请参阅“**KitProg3 用户指南**”。

PSoC™ Creator 支持一次调试单个 CPU (Cortex®-M4 或 Cortex®-M0+)。某些第三方 IDE 支持多 CPU 调试。有关使用 PSoC™ Creator 在 PSoC™器件上调试固件的更多信息，请参考 PSoC™ Creator 帮助。

2.6　　PSoC™ 6 MCU 开发套件

PSoC™ 6 Wi-Fi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT)和 **PSoC™ 6 Wi-Fi Bluetooth® Prototyping Kit (CY8CPROTO-062-4343W)**是开发套件，支持 PSoC™ 62 系列 MCU 以及 Wi-Fi 和 Bluetooth®连接。

PSoC™ 6 Bluetooth® LE Pioneer Kit (CY8CKIT-062-BLE) 和 **PSoC™ 6 Bluetooth® LE Prototyping Kit (CY8CPROTO-063-BLE)**支持具有蓝牙低功耗（Bluetooth® LE）连接功能的 PSoC™ 6 MCU。 有关更多信息，请参见 **PSoC™ 6 MCU 产品页面**。

3 器件特性

PSoC™ 6 MCU 器件具有广泛的功能集，如 **Figure 4** 所示。以下是其主要功能列表。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。

- **MCU 子系统**
  - 150-MHz Arm® Cortex®-M4 和 100-MHz Arm® Cortex®-M0+
  - 高达 2 MB 闪存，另外 32 KB 用于 EEPROM 仿真和 32 KB 监控闪存
  - 高达 1 MB 的 SRAM，可选择深度睡眠保持粒度，32 KB 保留边界
  - 硬件支持处理器间通信
  - DMA 控制器
- **安全特性**
  - 密码加速器和真随机数发生器功能
  - 一次性可编程 eFUSE，用于安全密钥存储
  - 使用基于硬件散列的身份验证进行安全启动
- **I/O 子系统**
  - 多达 104 个 GPIO，具有可编程驱动模式、驱动强度、压摆率
  - 两个带 Smart I/O 的端口，可以实现布尔运算
- **可编程模拟模块**
  - 两个 6 MHz 增益带宽 (GBW) 的运算放大器和两个低功耗比较器
  - 最多一个 12 位，1 Msps SAR ADC 和一个 12 位电压模式 DAC
- **可编程数字模块，通信接口**
  - 最多 12 个 UDB 用于定制数字外设
  - 最多 32 个 TCPWM 模块可配置为 16 位/32 位定时器，计数器，PWM 或正交解码器
  - 最多 13 个 SCB 可配置为 I2C 主设备或从设备，SPI 主设备或从设备或 UART
  - 控制器区域网络接口，具有灵活的数据速率
  - 最多两个安全数字主机控制器，支持 SD、SDIO 和 eMMC 接口
  - 具有最多两个 I2S 接口和两个 PDM 通道的音频子系统
  - SMIF 接口，支持从外部四路 SPI 闪存执行就地和即时加密和解密
  - 具有设备和主机功能的全速，双角色 USB
- **带有 SmartSense 自动调谐功能的 CAPSENSE™**
  - 支持 CAPSENSE™ Sigma-Delta (CSD) 和 CAPSENSE™发送/接收 (CSX) 控制器
  - 提供同类最佳的 SNR，液体容差和接近感应
- **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，用户可在 1.1 V 或 0.9 V 范围内选择内核逻辑工作电压
  - 多个片上稳压器: 低压差 (活动，深度休眠模式下的 LDO)，降压转换器
  - 六个功耗模式用于精细化功率管理
  - 具有内置 RTC，电源管理集成电路 (PMIC) 控制和有限 SRAM 备份的“始终开启”备用电源域

器件特性

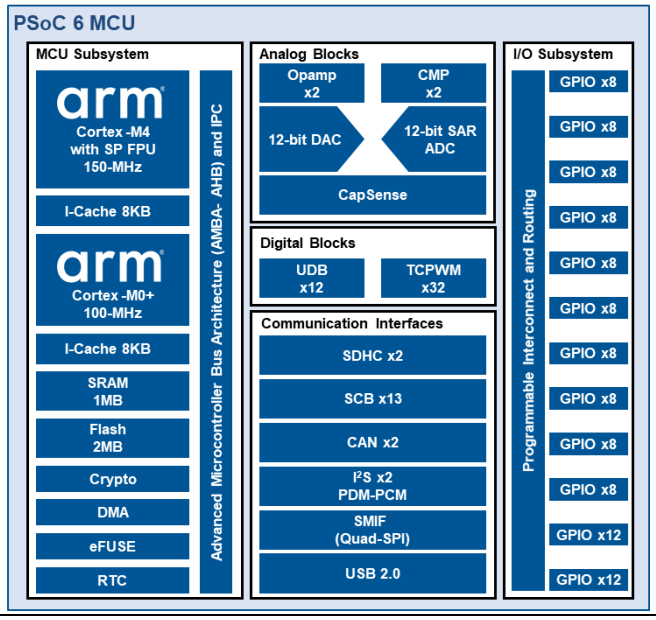

Figure 4 PSoC™ 6 MCU 框图

4 我的首个使用 PSoC™ Creator 的 PSoC™ 6 MCU 设计

本节执行以下操作:

- 演示如何构建基于 PSoC™ 6 MCU 的简单设计，并将其编程到开发套件中。
- 提供详细步骤，使您可以轻松学习 PSoC™ 6 MCU 设计技术以及如何使用 PSoC™ Creator IDE。

4.1 使用这些说明

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

- **第 1 部分: 从零开始创建新项目**
- **第 2 部分: 实施设计**
- **第 3 部分: 生成源代码**
- **第 4 部分: 编写固件**
- **第 5 部分: 构建项目并对设备进行编程**
- **第 6 部分: 测试您的设计**

如果您熟悉使用 PSoC™ Creator 开发项目，则可以直接使用代码示例 **CE221773 – PSoC™ 6 MCU Hello World Example** 的 PSoC™ Creator 版本。它是一个完整的设计，编写了所有固件。您可以阅读说明并观察代码示例中的步骤是如何实现的。

如果您从头开始并按照本应用笔记中的所有说明操作，则在遵循说明的同时使用代码示例作为参考。

您可以点击上面的链接从英飞凌网站下载代码示例。您也可以使用 PSoC™ Creator 中 **File** > **Code Example** 命令。将器件系列设置为 PSoC™ 62。选择 PSoC™ MCU Hello World 示例。通过单击示例旁边的下载图标下载代码示例，然后单击“**Create Project**”，并按照屏幕上的说明进行操作。

该设计是为 **CY8CKIT-062-WiFi-BT PSoC™ 6 Wi-Fi-BT Pioneer Kit** 套件开发的。您也可以使用 **CY8CKIT-062-BLE PSoC™ 6 Bluetooth® LE Pioneer Kit** 套件通过从器件选择器中选择适当的器件来测试该示例。

4.2 关于设计

该设计使用 PSoC™ 6 MCU 的 CM4 CPU 执行两项任务: UART 通信和 LED 控制。在器件复位时，CM0+ CPU 使能 CM4 CPU。CM4 CPU 使用 UART 组件向串行端口流打印“Hello World”消息，当用户按下 Enter 键时，PSoC™ 6 MCU Wi-Fi-BT Pioneer Kit 上的 LED 开始闪烁。

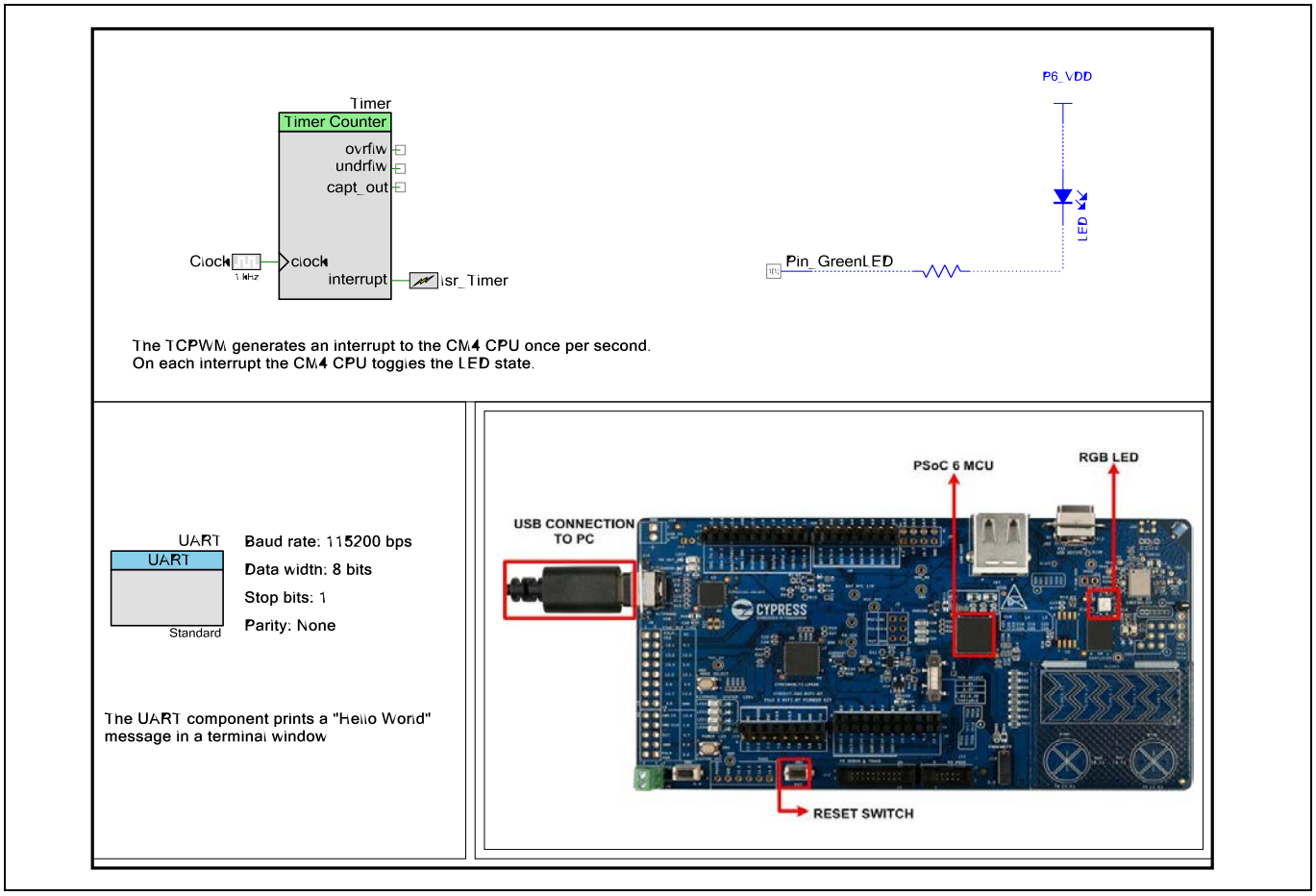

Figure 5　我的第一个 PSoC™ 6 MCU 设计

4.3　第 1 部分: 从零开始创建新项目

本节将带您逐步了解设计过程。它首先创建一个空项目，并指导您完成硬件和固件设计开发阶段。

Note:　*这些说明假定您使用的是 PSoC ™Creator 4.2。对于后续版本的 PSoC ™Creator，整个开发过程是相同的; 但是用户界面可能会随着时间而改变。*

启动 PSoC™ Creator 并开始使用。

1. 确保 PSoC™ Creator 可以找到 PDL。

   这应该在安装过程中自动正确设置，但如果没有正确设置则无效。有关此步骤的帮助，请参见 **Figure 6**。

   a) 选择 **Tools** > **Options**。

   b) 在 **Project Management** 面板上，检查 **PDL v3 (PSoC™ 6 Devices) location** 字段中的路径。

   c) 确保它是正确的。如果不是，请单击 **Browse** 按钮，找到 PDL 的已安装目录。默认位置是 *C:\Program Files (x86)\Cypress\PDL\3.0.1*。

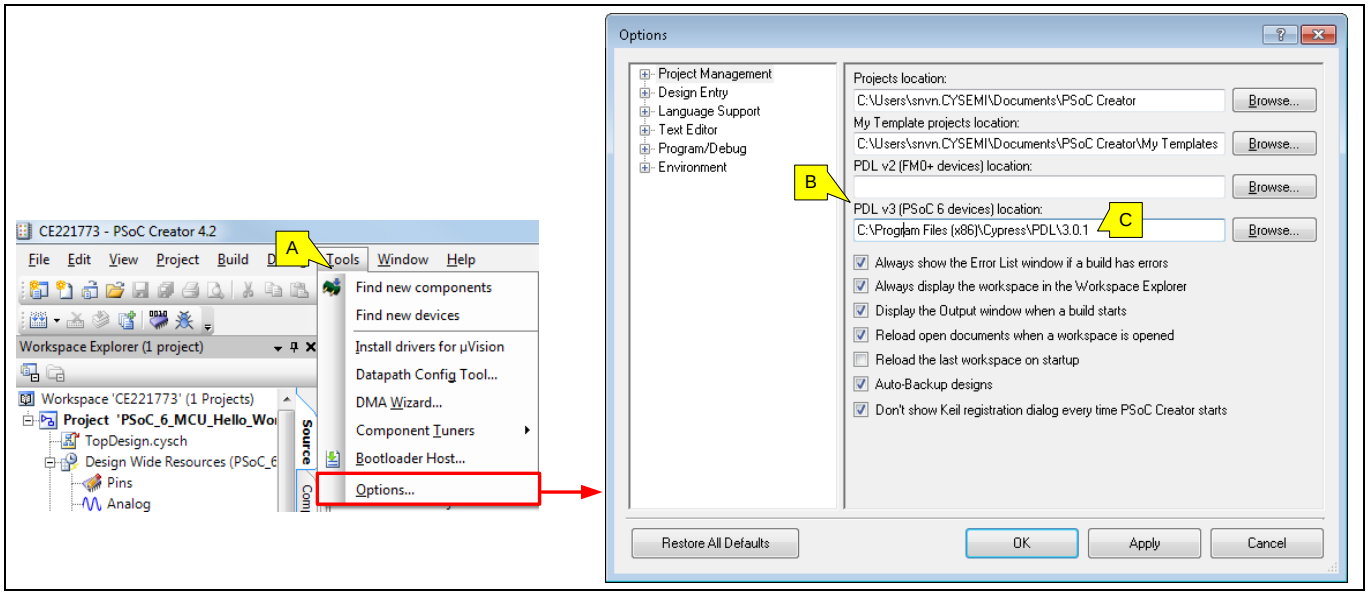

Figure 6　外设驱动程序库 (PDL) 位置

可选: 跳到**第 2 部分: 实施设计**

2. 创建新 PSoC™ Creator 项目

选择 **File** > **New** > **Project**，如 **Figure 7** 所示。将出现 **Create Project** 窗口。

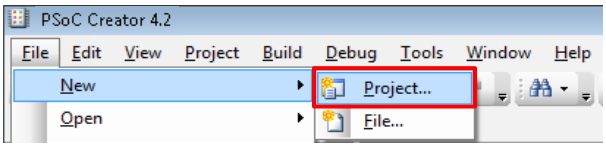

Figure 7　创建一个新的 PSoC™ Creator 项目

Note:　*如果您使用的是代码示例，请选择 **File** > **Open** > **Project/Workspace**，如 **Figure 8** 所示。出现 **Open** 窗口。指向代码示例工作空间的位置并打开工作空间。*

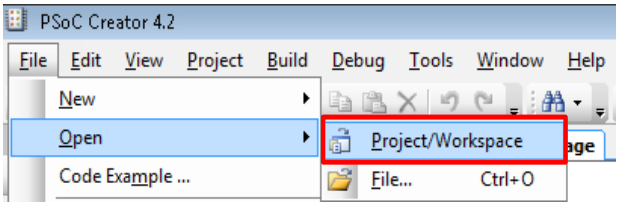

Figure 8　打开现有代码示例工作区

3. 选择 PSoC™ 6 MCU 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来加速开发过程。有关此步骤的帮助，请参见 **Figure 9**

a) 单击目标设备。

b) 在系列下拉菜单中，选择 **PSoC 6**。

c) 在设备下拉菜单中，选择 **PSoC 62**。

d) 单击下一步。将出现“选择项目模板”面板。

PSoC™ Creator 使用 CY8C6247BZI-D54 作为 PSoC™ 6 MCU 系列的默认器件。该器件安装在 **CY8CKIT-062-WiFi-BT PSoC™ 6 Wi-Fi-BT Pioneer Kit** 套件上。

如果您使用的是基于 PSoC™ 6 MCU 的自定义硬件或不同的 PSoC™ 6 MCU 部件号，则可以选择在目标设备中选择启动设备选择器选项并选择相应的部件号。

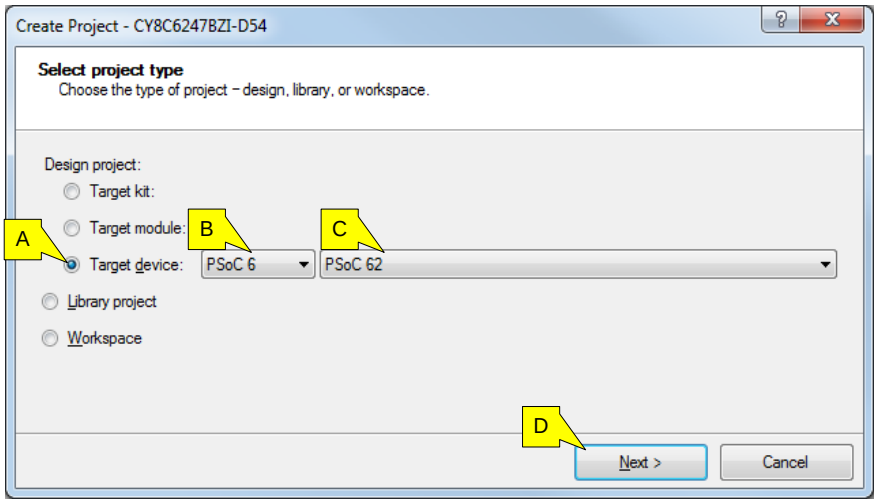

Figure 9　选择目标设备

4. 选择项目模板

a) 选择 **Empty Schematic**

b) 单击 **Next**

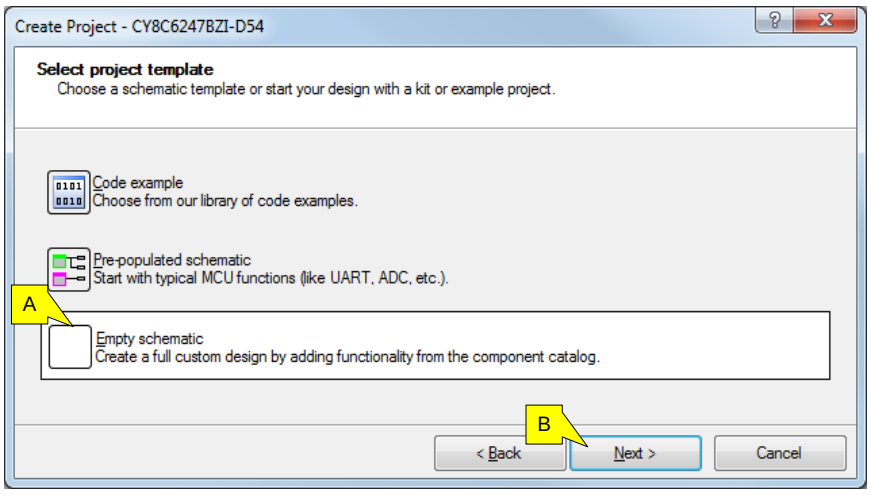

Figure 10　选择项目模板

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，禁用所有导出选项。如果情况发生变化，您可以稍后修改此设置。

单击“**Next**”接受默认选项。

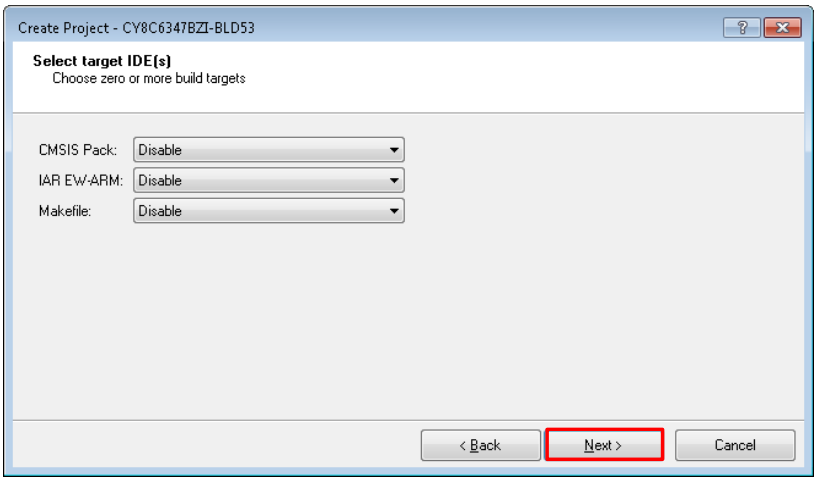

Figure 11　选择目标 IDE (全部禁用)

6. 创建项目

在此步骤中，您可以设置工作区的名称和位置，以及项目的名称。有关此步骤的帮助，请参见 **Figure 12**。工作空间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 **Project Name**。项目和工作区名称可以相同或不同。

d) 单击 **Finish**。

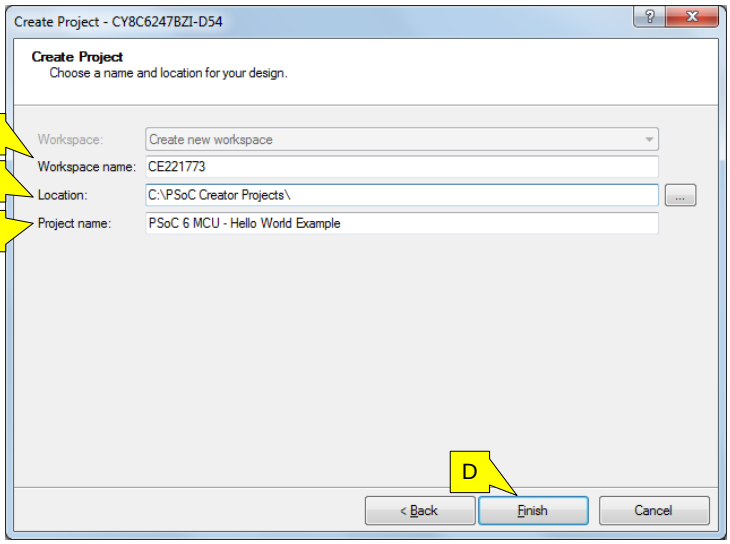

Figure 12　项目命名和位置

您已成功创建新的 PSoC™ Creator 项目。

4.4 第 2 部分: 实施设计

现在您有了一个项目文件，现在是时候使用 PSoC™ Creator 组件实现硬件设计了。如果您直接使用代码示例，则您已经拥有完整的设计。

在实现设计之前，需要快速浏览 PSoC™ Creator 接口。

如 **Figure 13** 所示，在 PSoC™ Creator 应用程序中显示空的设计原理图。

该项目包括一个带有基本文件集的项目文件夹。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，随时打开原理图。在新项目中，原理图为空。如果您使用的是代码示例，那么这就是设计的原理图。

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...**”文本框中键入组件的名称然后按 Enter 键来搜索特定组件。见 **Figure 13**。

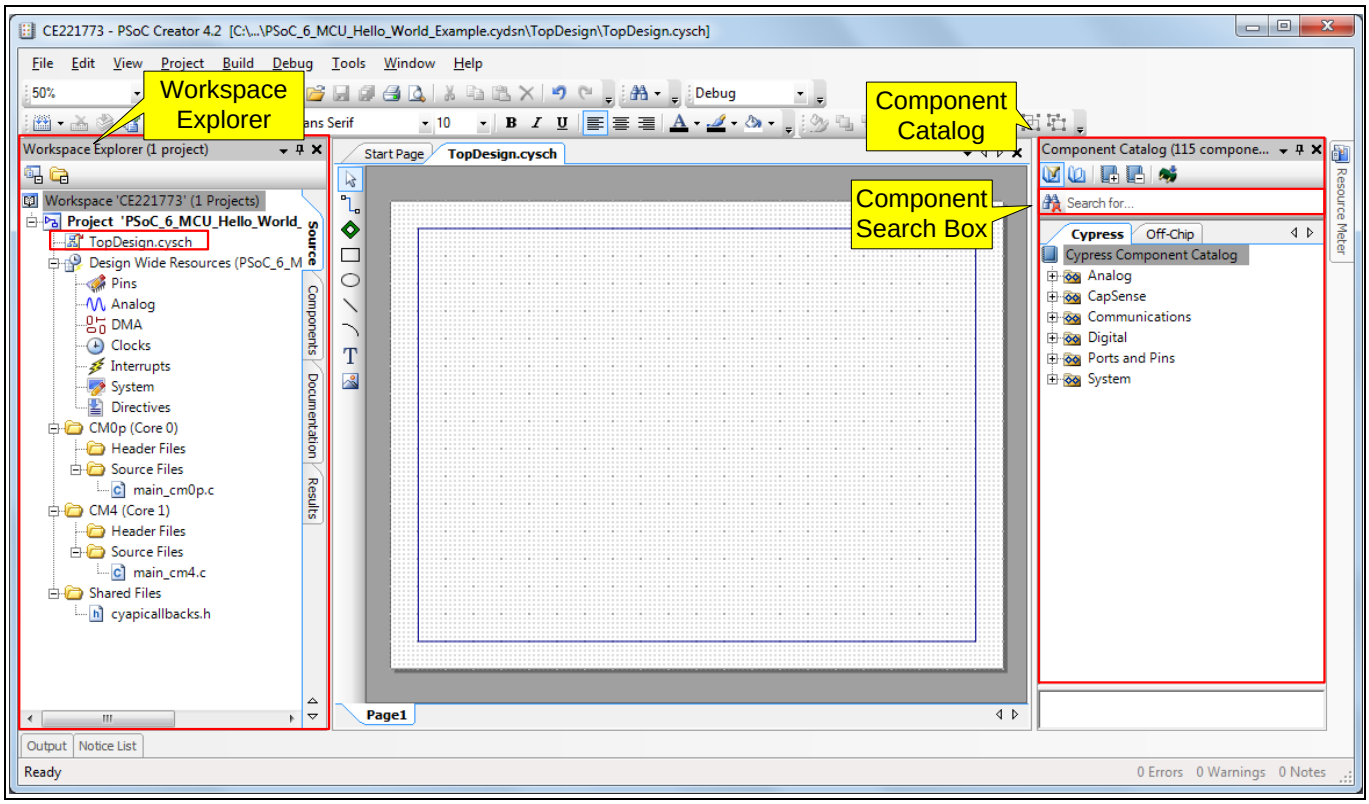

Figure 13 原理图和组件目录

1. 在设计中放置组件

   该设计使用多个组件: 三个数字输出引脚，一个 UART，一个看门狗定时器和一个中断。在此步骤中，您将它们添加到设计中。您可以在后续步骤中配置它们。**Figure 14** 显示了结果。

   a) 在组件目录中，展开“**Communications**”组，将 UART (SCB) 组件拖到原理图中，然后放下。放置组件的位置无关紧要。

   b) 展开“**Ports and Pins**”组，然后将数字输出引脚拖入设计中。

   c) 展开数字组，并将计时器计数器 (TCPWM) 组件拖入设计中。

   d) 展开“**System**”组，然后将“**Interrupt** 组件”和“**Clock** 组件”拖到设计中。

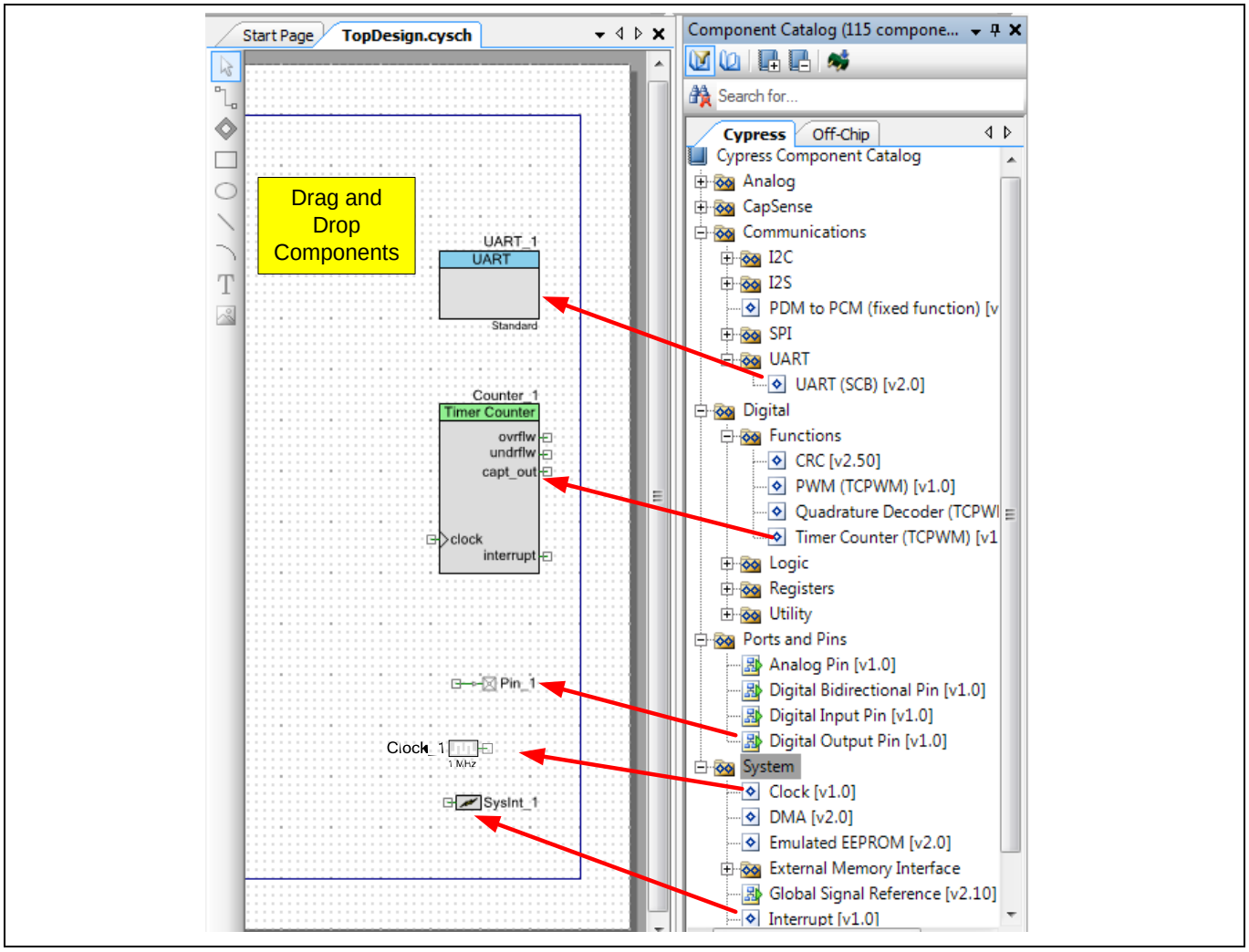

Figure 14　在设计中放置组件

PSoC™ Creator 为每个组件提供默认名称和属性。默认值可能适用于任何给定设计，也可能不适合。在后续步骤中，您将修改名称和某些属性。

2. 配置 LED 引脚

输出引脚驱动 LED。PSoC™ 6 Wi-Fi-BT Pioneer 套件上的 LED 为低电平有效; 也就是说，逻辑 HIGH 引脚驱动状态将 LED 关闭，逻辑低引脚驱动状态将其打开。**Figure 15** 显示了配置。

双击原理图上放置的组件以打开配置对话框。然后执行以下步骤。

a) 实例的名称更改为 **Pin_GreenLED**。

b) 将组件取消选择硬件连接。固件将驱动引脚。

c) 将驱动模式设置为电阻上拉。

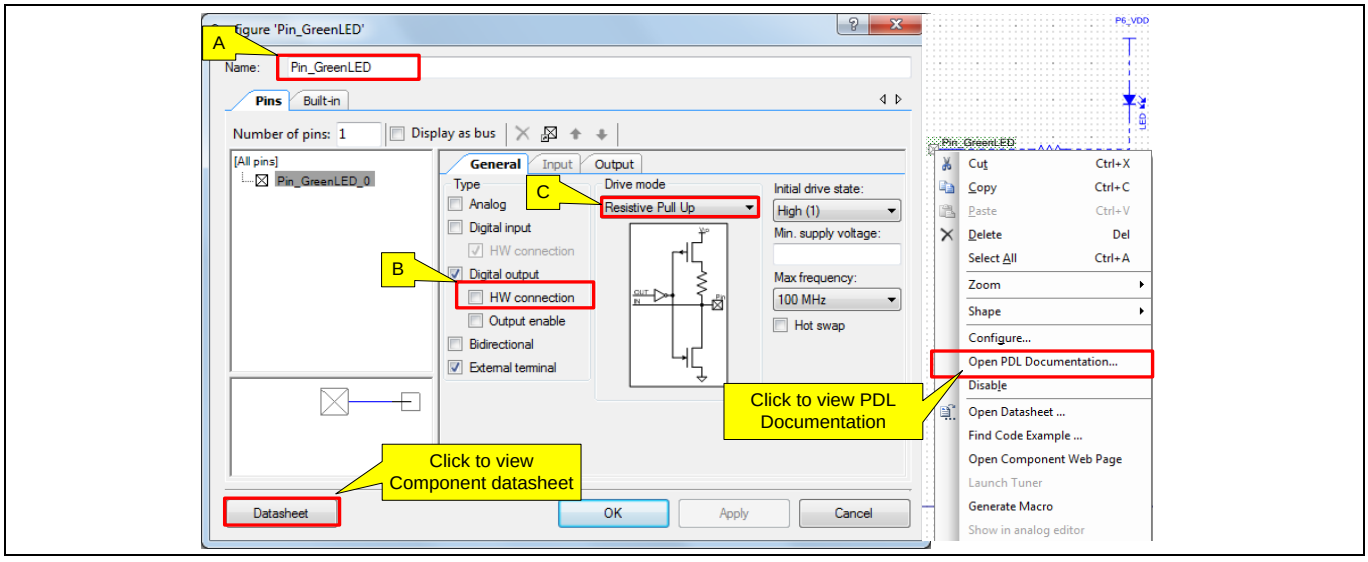

Figure 15　配置输出引脚组件

Note:

1. *每个组件都有一个关联的数据表，可以从配置窗口访问。组件数据表提供了有关组件配置、应用程序编程接口(API) 和电气规范的更多信息。*
2. *通过右键单击组件并单击“**Open PDL Documentation...**”链接，可以打开组件的关联 PDL 驱动程序的 API 参考文档。见 **Figure 15**。*
3. *对于引脚，如果启用外部端子，则可以将外部“片外”组件添加到设计中。原理图上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。您还可以向设计添加描述文本框。**Figure 16** 显示了如何增强 LED 的设计。在这种情况下，片外组件配置为未选中 **Instance_Name_Visible** 选项。电阻配置为值字段留空。配置电源端子，设置 **Supply_Name** 为 P6_VDD。*

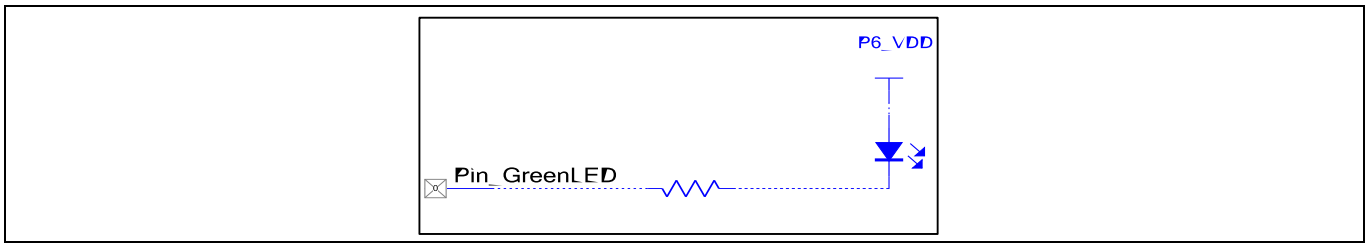

Figure 16　具有片外组件的输出引脚

3. 配置 UART 组件

双击组件以打开配置窗口。该设计使用此组件在终端窗口中以 115200 bps 的波特率显示消息。

a) 将组件实例的名称更改为 UART。

b) 单击“**OK**”。

对于所有其他设置，该设计使用默认值。

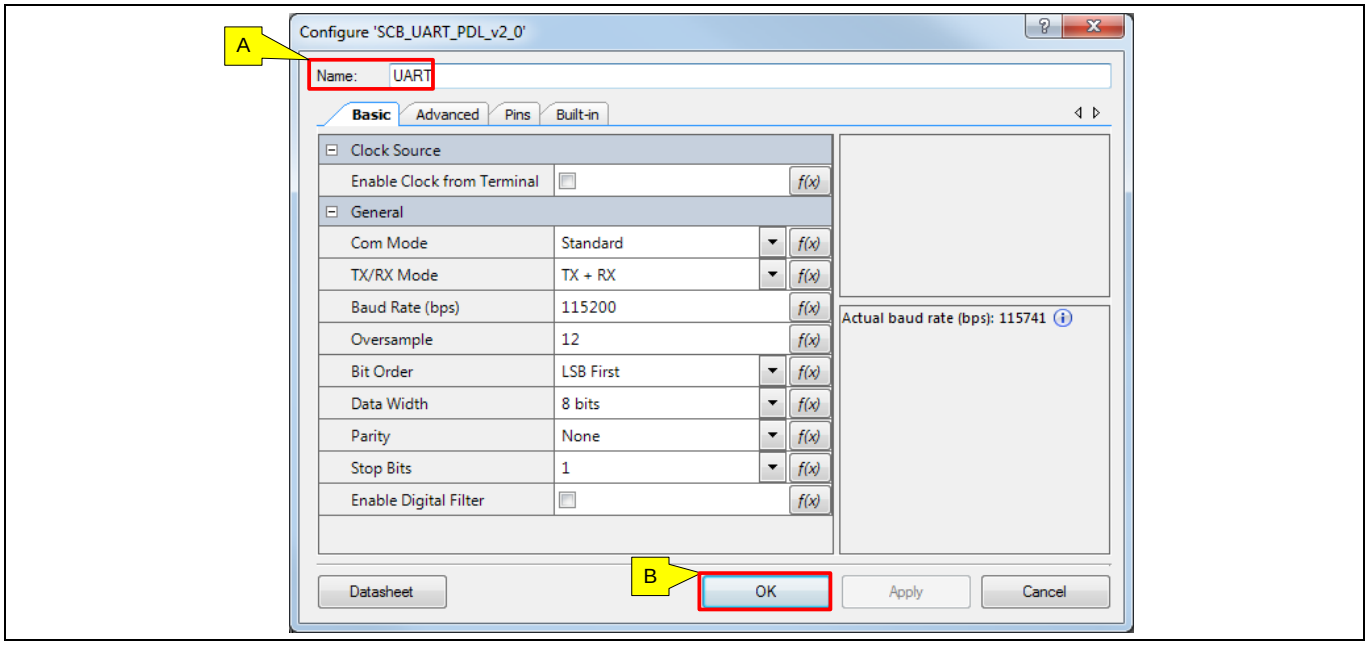

Figure 17　配置基于 SCB 的 UART 组件

4. 配置定时器计数器 (TCPWM) 组件以触发中断

在此步骤中，将 Timer (TCPWM) 组件配置为每秒触发一次中断 (1 Hz)。TCPWM 的时钟源是外设时钟 (Clk_Peri)。该设计将使用此中断来切换 LED 状态。打开组件自定义器，然后按照 **Figure 18** 中所示的步骤操作。

a) 将 **Name** 更改为 **Timer**。

b) 将 Period 设置为 1000，将 Interrupt Source 设置为 Overflow / Underflow。

c) 单击 **OK** 完成 TCPWM 组件的配置。

d) 将 TCPWM 的时钟端子连接到 1 kHz 时钟源。在原理图中，使用线工具按钮或按“W”键开始将时钟组件连接到 TCPWM 组件的时钟端子。

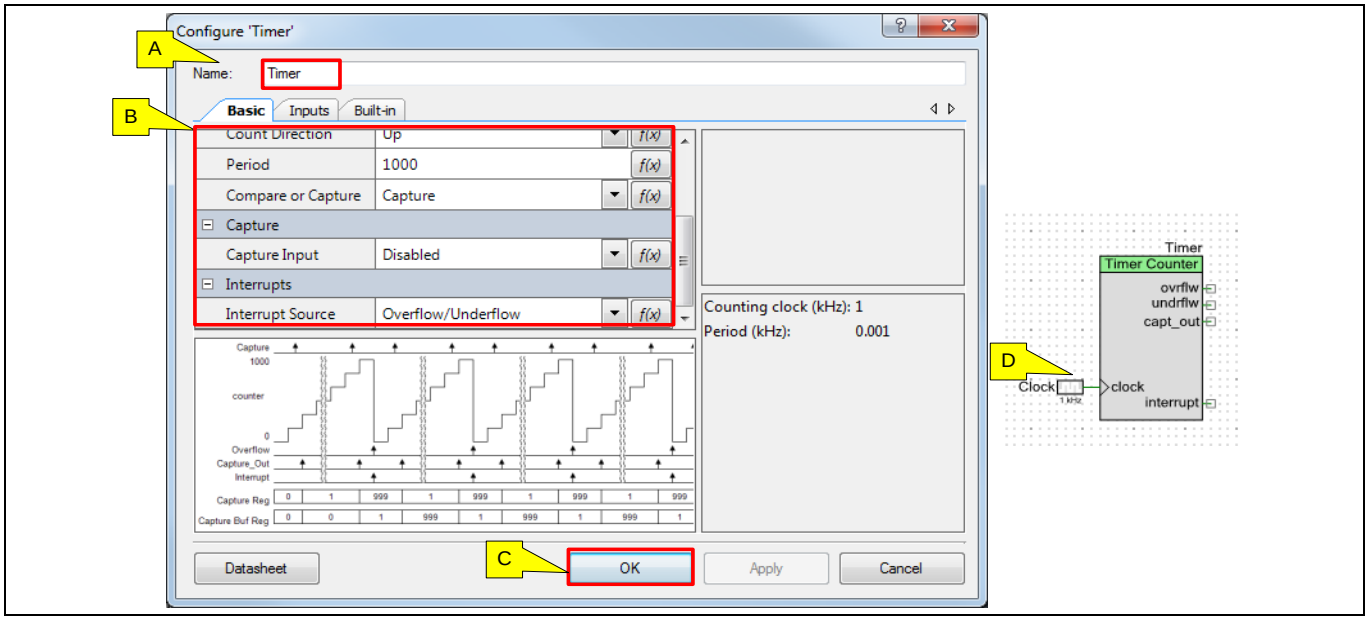

Figure 18　配置 TCPWM 组件

5. 配置中断组件

在此步骤中，您将配置 SysInt 组件以将 TCPWM 中断映射到 CM4 CPU。打开组件自定义程序，然后按照 **Figure 19** 中所示的步骤操作。

a) 将 **Name** 更改为 **Isr_Timer**

b) 单击“**OK**”完成 SysInt 组件的配置

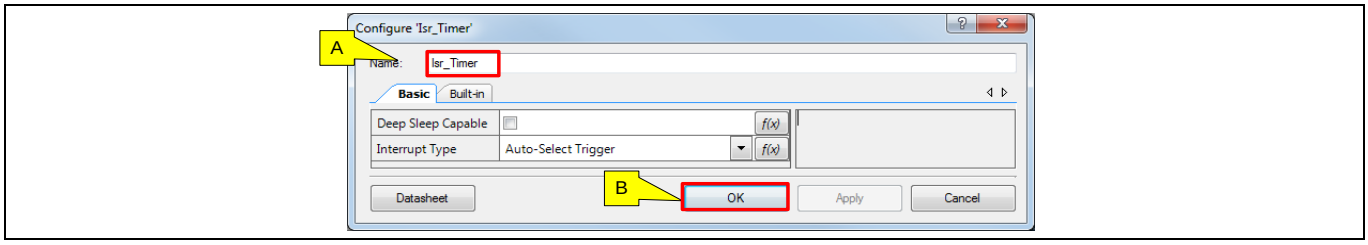

Figure 19　SysInt_PDL 设置

最后一步，将 TCPWM 组件的中断输出连接到 Isr_TCPWM 组件输入。这会将 TCPWM 中断路由到 CM4 CPU (此中断的 CM4 CPU 选择将在后续步骤中的系统中断配置中设置)。在原理图中，使用线工具按钮或按“W”键开始连接组件。

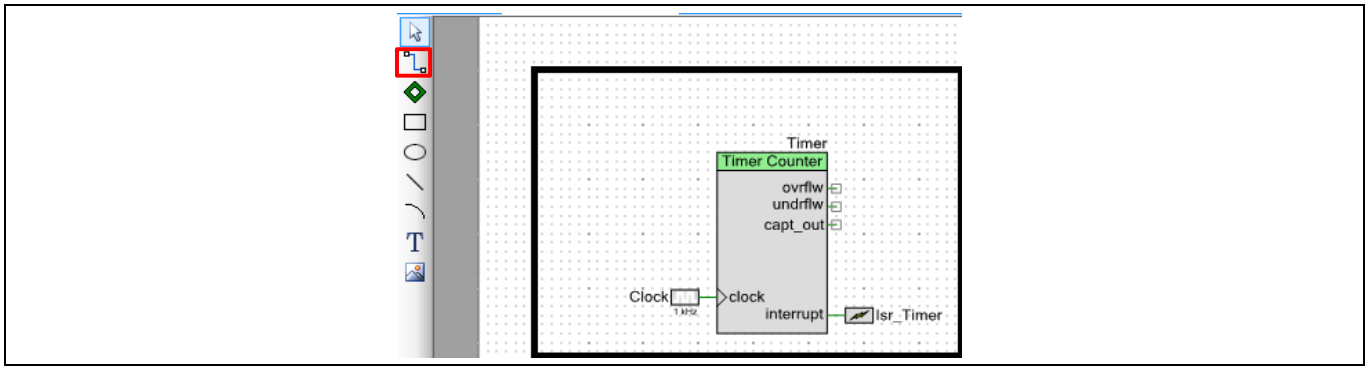

Figure 20　将 TCPWM 外设中断连接到 CM4 CPU

6. 设置每个引脚组件的物理引脚

完成设计还有一项任务。您必须将每个组件与设备上所需的物理引脚相关联。选择使用哪个引脚是由电路板设计驱动的。您可以在套件原理图中找到此信息。**Figure 21** 显示了该步骤的结果。您可以将外部 LED 连接到所选引脚。

要设置引脚，请在相应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，因为这些名称与所使用的特定包无关。

a) 打开引脚选择器
在“**Workspace Explorer**”窗格中，双击“Design Wide Resources”下的“**Pins**”项。出现该设备的引脚选择器。

b) 设置每个引脚，如 **Table 1** 所示。

Table 1　　CY8CKIT-062-WiFi-BT Pioneer Kit 套件的物理引脚分配

引脚组件名称	端口名
UART: rx	P5[0]
UART: tx	P5[1]
Pin_GreenLED	P1[1]

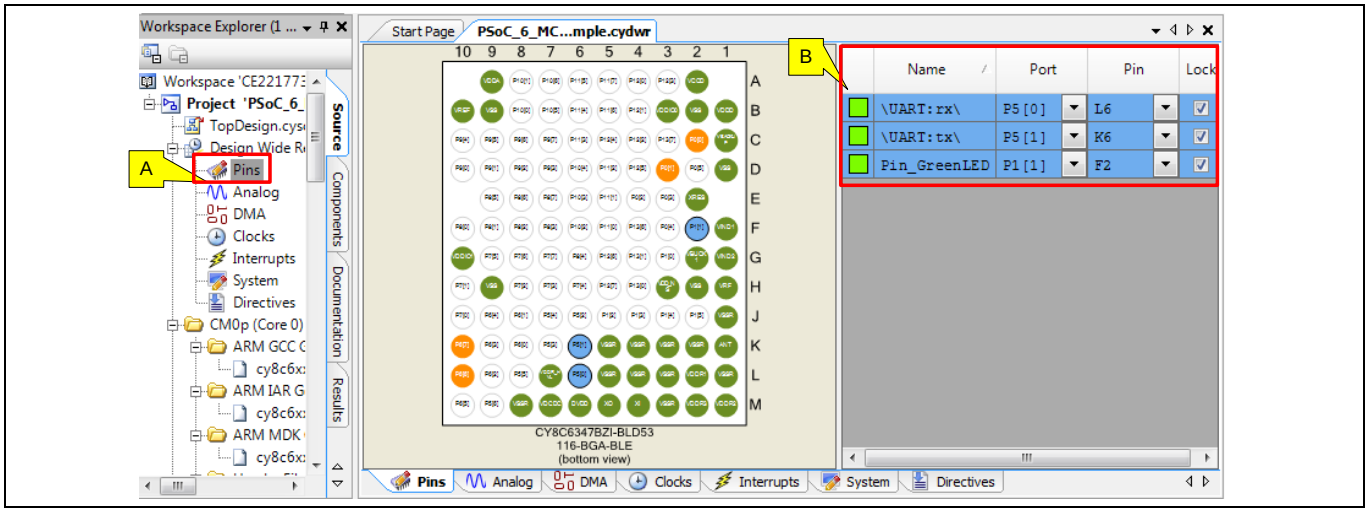

Figure 21　引脚分配

7. 配置系统时钟

该设计使用高频系统时钟设置的默认值。虽然您不修改此设计的高频时钟，但您应该了解 PSoC™ Creator 如何管理它们。如果您使用自己的电路板，则可能需要修改这些时钟。

a) 在 Workspace Explorer 窗格中，双击 Design Wide Resources 下的 Clocks 项。出现时钟列表。

b) 单击编辑时钟。出现配置系统时钟对话框。
在这里，您可以看到时钟树，并根据需要修改时钟。请注意，有不同类型时钟的选项卡，**如源时钟**，**FLL/PLL**，**高频时钟和**其他时钟。

c) 单击 **FLL/PLL** 选项卡。默认情况下，**PSoC™ Creator 启用 FLL** 并将频率设置为 100 MHz。

d) 单击“**High Frequency Clocks**”选项卡。

e) 您可以通过在 **Clk_Fast** 中设置分频器来设置 CM4 CPU 时钟。默认情况下，分频器设置为 1。

f) 您可以通过在 **Clk_Slow** 中设置分频器来设置 CM0+ CPU 时钟。默认情况下，分频器设置为 1。请参见 **Figure 22**。

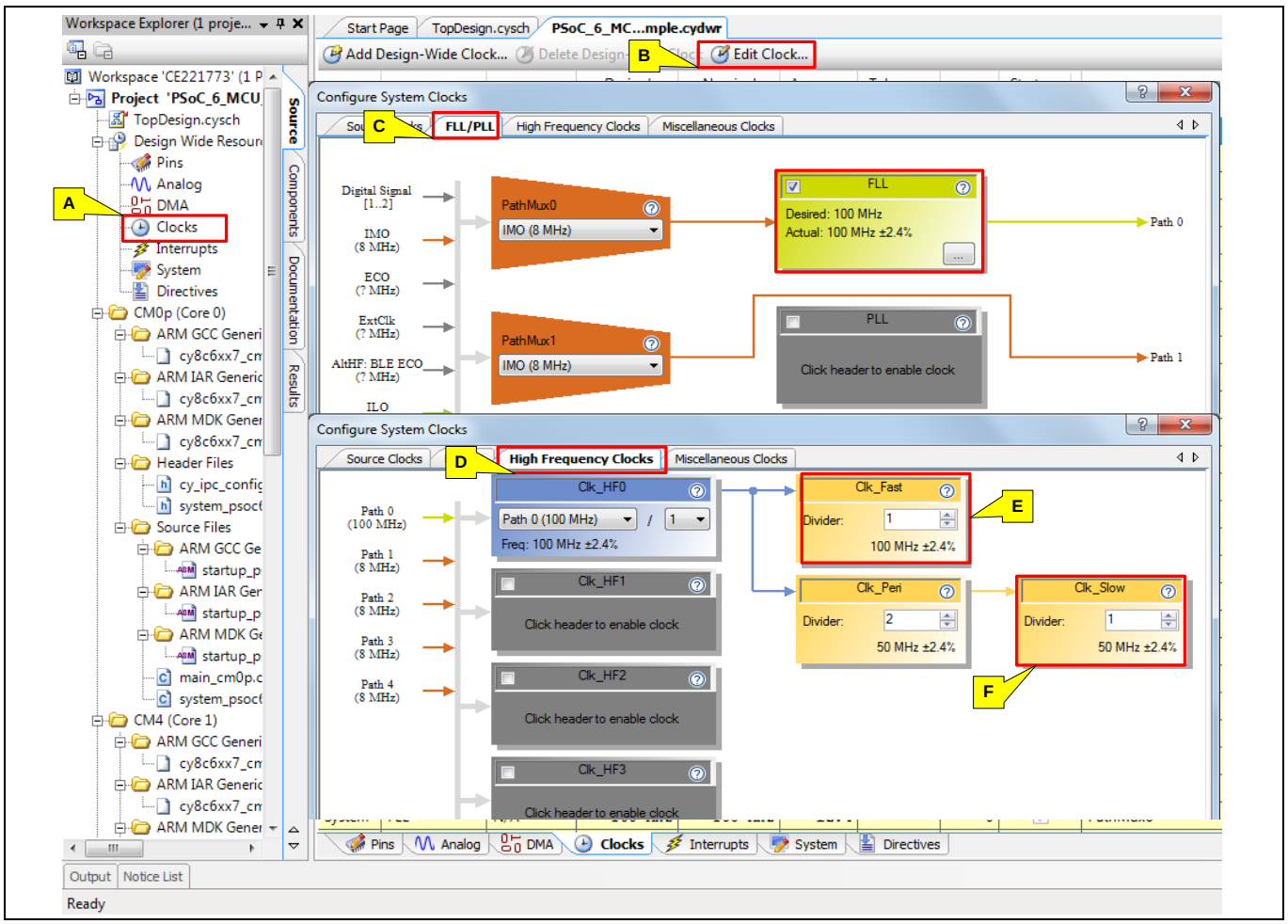

Figure 22　时钟配置

8. 配置系统中断

在此步骤中，您将配置系统中断。见 **Figure 23**。

a) 在 **Workspace Explorer** 窗格中，双击 **Design Wide Resources** 下的 **Interrupts** 项。出现中断列表。

b) 为 CM4 CPU 启用 **Isr_Timer**。

在**第 3 部分: 生成源代码**中生成代码时，PSoC™ Creator 会自动生成中断号。

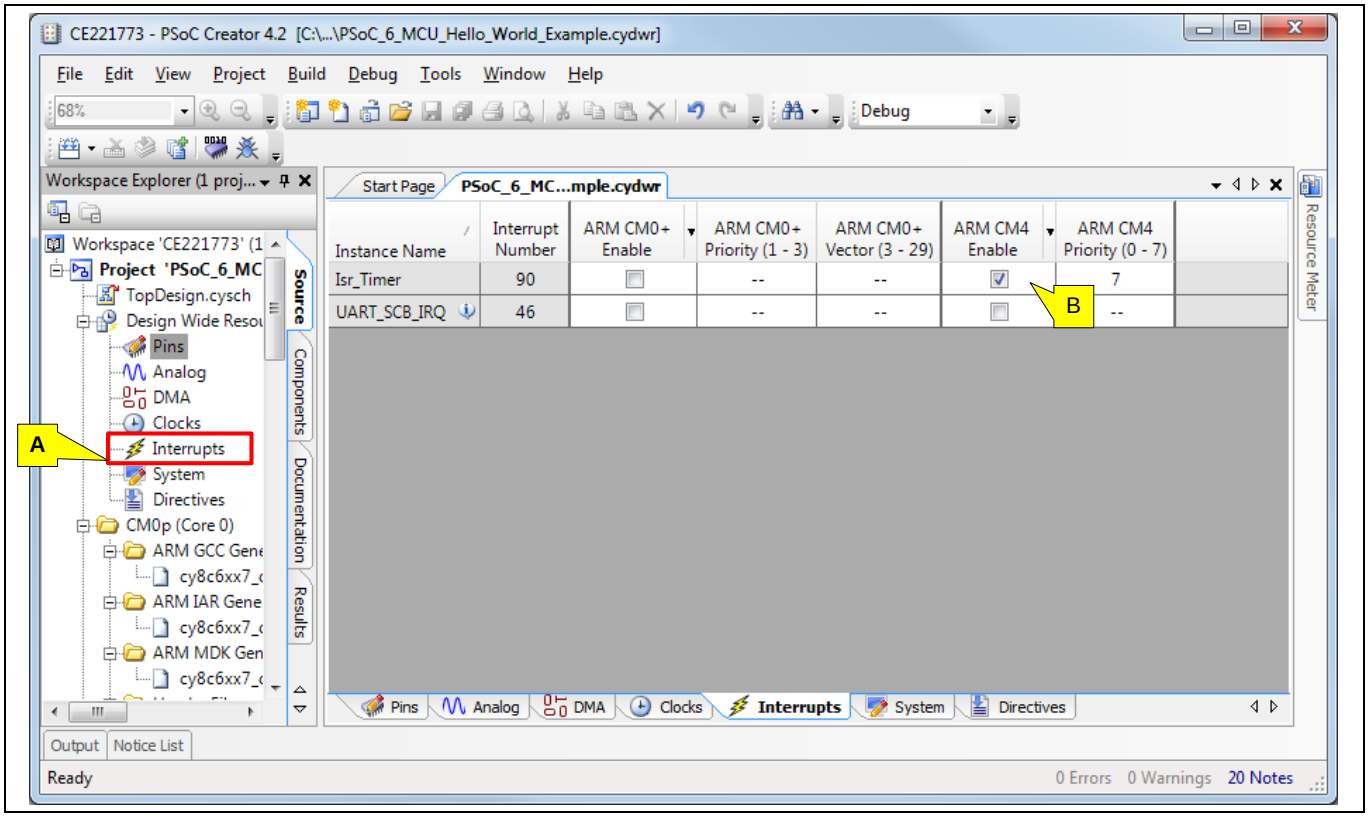

Figure 23 中断配置

开发过程的下一部分是生成代码。

Note: 本练习未详细说明如何将工作导出到目标 IDE。但是，如果您希望使用目标 IDE，那么在生成源代码之前，您需要确保选择正确的目标 IDE。

4.5 第 3 部分: 生成源代码

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

1. 生成应用程序

   选择 **Build** > **Generate Application**。PSoC™ Creator 根据设计生成源代码，并将文件放在 Generated_Source 文件夹中。见 **Figure 24**。PSoC™ Creator 会提醒您可能出现的错误或问题。如果您从头开始工作并遇到错误，请重新阅读**第 2 部分: 实施设计**的配置步骤，以确保您已正确执行它们。

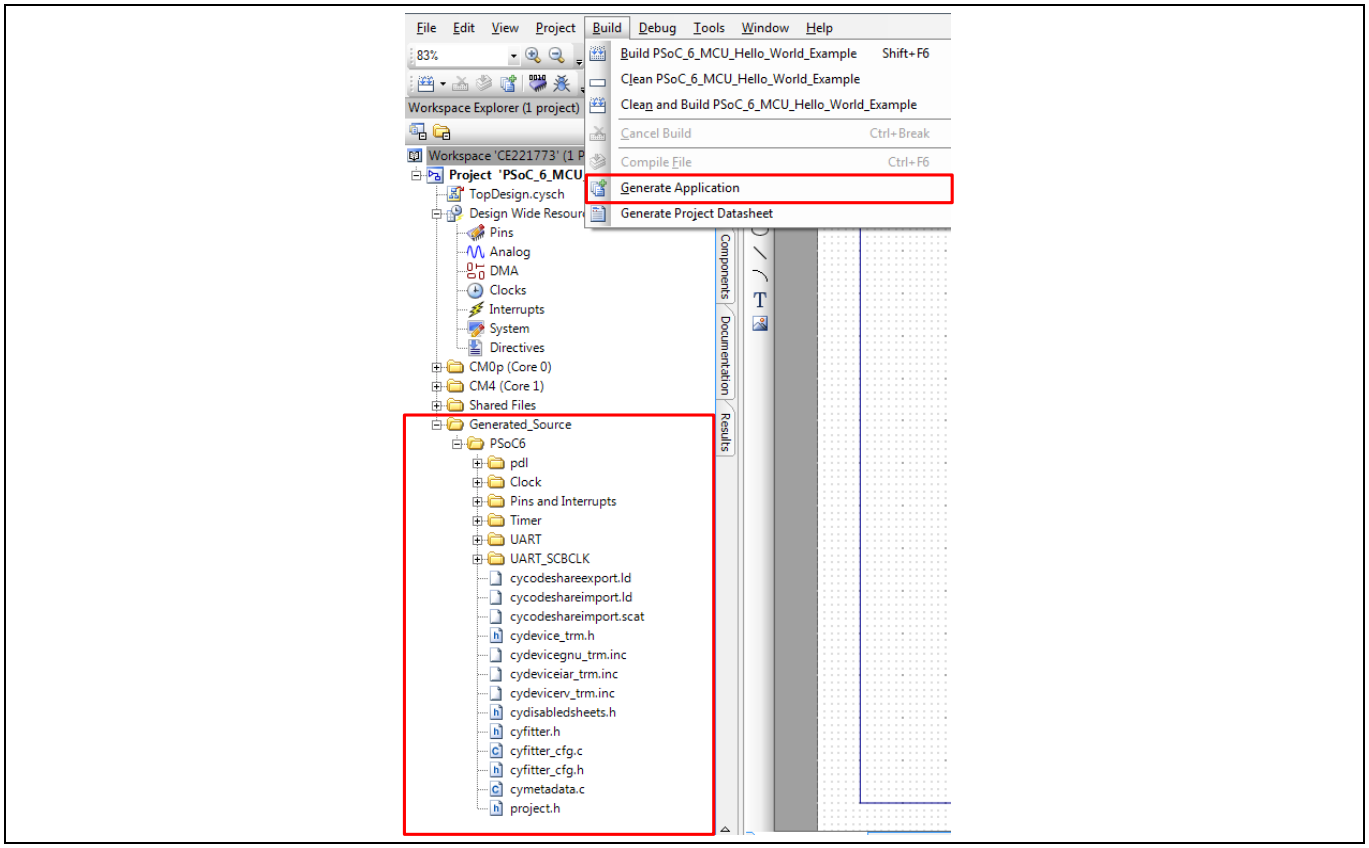

Figure 24 生成应用程序

背景: PSoC™ 6 MCU 是一款双 CPU 平台。您可以将固件目标设置为在 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此项。右键单击源文件，然后选择 **Properties**。**Figure 25** 显示了 **Properties** 对话框窗口。默认情况下，*main_cm0p.c* 文件的目标是 Cortex®-M0+，而 *main_cm4.c* 文件的目标是 Cortex®-M4。您无需修改任何其他文件的属性。它们已在代码示例中设置。

按照惯例，目标在 CM0+ CPU 上运行的文件位于 *CM0p* 文件夹中，而目标在 CM4 CPU 上运行的文件位于 *CM4* 文件夹中。

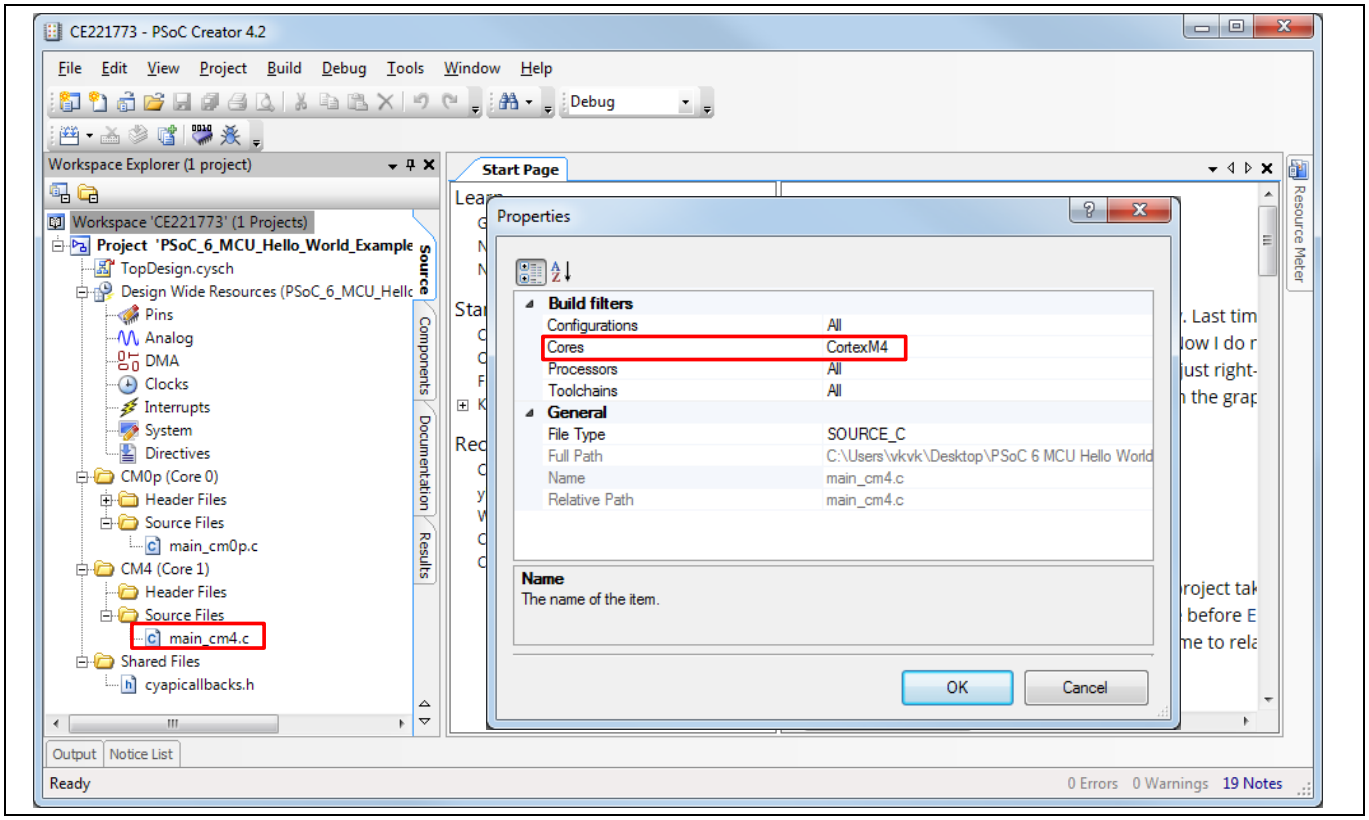

Figure 25 设置源 C 文件的目标处理器

4.6 第 4 部分: 编写固件

在开发过程的这一点上，您已经创建了一个项目，实现了硬件设计并生成了代码。在本部分中，您将编写实现设计功能的固件。

本部分中的步骤将讨论您在**第 2 部分: 实施设计**中配置的设计的固件。

代码示例包含所有必需的代码。如果您是从头开始工作，则可以从本节提供的代码段中将相应的源代码复制到 *main_cm0p.c* 和 *main_cm4.c*。如果您使用的是代码示例，则文件已在您的项目中。

固件流程

在剩下的步骤中，我们检查了 *main_cm0p.c* 和 *main_cm4.c* 文件中的代码。

当 PSoC™ 6 MCU 器件复位时，固件首先执行系统初始化，包括设置 CPU 执行，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 之间分开。CM0+ CPU 退出复位并启用 CM4 CPU。下面给出了 CM0+ CPU 代码片段。将以下代码段复制到项目的 *main_cm0p.c* 文件中。

Code Listing 1

```
/* Header files includes*/
#include "project.h"
int main(void)
{
    __enable_irq(); /* Enable global interrupts. */
```

Code Listing 1

```
    /* Enable CM4.  CY_CORTEX_M4_APPL_ADDR must be updated
       if CM4 memory layout is changed. */
    Cy_SysEnableCM4(CY_CORTEX_M4_APPL_ADDR);

    for(;;)
    {

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

启用 CM4 CPU 后，UART 组件将启动并在终端仿真器上打印“Hello World！”消息。定时器计数器 PWM (TCPWM) 组件配置为每秒产生一次中断。在每次中断时，CM4 CPU 切换套件上的 LED (**LED5**) 状态。将以下代码段复制到项目的 *main_cm4.c* 中。

Code Listing 2

```
/* Header files includes*/
#include "project.h"
/*******************************************************************************
* Macros
*******************************************************************************/
#define LED_ON    (0)
#define LED_OFF   (!LED_ON)
/*******************************************************************************
* Function Prototypes
*******************************************************************************/
void UartInit(void);
void TimerInit(void);
void Isr_Timer(void);

/*******************************************************************************
* Global Variables
*******************************************************************************/
bool LEDupdateFlag = false;
/*******************************************************************************
* Function Name: main
*******************************************************************************/
int main(void)
{
    /* Start the UART peripheral */
    UartInit();

    /* Enable global interrupts. */
    __enable_irq();

    /* \x1b[2J\x1b[;H - ANSI ESC sequence for clear screen */
    Cy_SCB_UART_PutString(UART_HW, "\x1b[2J\x1b[;H");
```

Code Listing 2

```
    Cy_SCB_UART_PutString(UART_HW, "******************CE221773 - PSoC 6 MCU:"\
                  " Hello World! Example******************\r\n\n");

    Cy_SCB_UART_PutString(UART_HW, "Hello World!!!\r\n\n");

    Cy_SCB_UART_PutString(UART_HW, "Press Enter key to start blinking the LED\r\n\n");

    /* Wait for the user to Press Enter key */
    while(Cy_SCB_UART_Get(UART_HW) != '\r');

    /* Start the TCPWM peripheral. TCPWM is configured as a Timer */
    TimerInit();

    Cy_SCB_UART_PutString(UART_HW, "Observe the LED blinking on the kit!!!\r\n");

    for(;;)
    {
        if(LEDupdateFlag)
        {
            /* Clear the flag */
            LEDupdateFlag = false;

            /* Invert the LED state*/
            Cy_GPIO_Inv(Pin_GreenLED_0_PORT, Pin_GreenLED_0_NUM);
        }
    }
}
/*******************************************************************************
* Function Name: UartInit
*******************************************************************************/
void UartInit(void)
{
    /* Configure the UART peripheral.
       UART_config structure is defined by the UART_PDL component based on
       parameters entered in the Component configuration*/
    Cy_SCB_UART_Init(UART_HW, &UART_config, &UART_context);

    /* Enable the UART peripheral */
    Cy_SCB_UART_Enable(UART_HW);
}
/*******************************************************************************
* Function Name: TimerInit
*******************************************************************************/
void TimerInit(void)
{
    /* Configure the TCPWM peripheral.
       Counter_config structure is defined based on the parameters entered
       in the Component configuration */
```

Code Listing 2

```
    Cy_TCPWM_Counter_Init(Timer_HW, Timer_CNT_NUM, &Timer_config);

    /* Enable the initialized counter */
    Cy_TCPWM_Counter_Enable(Timer_HW, Timer_CNT_NUM);

    /* Start the enabled counter */
    Cy_TCPWM_TriggerStart(Timer_HW, Timer_CNT_MASK);

    /* Configure the ISR for the TCPWM peripheral*/
    Cy_SysInt_Init(&Isr_Timer_cfg, Isr_Timer);

    /* Enable interrupt in NVIC */
    NVIC_EnableIRQ((IRQn_Type)Isr_Timer_cfg.intrSrc);
}
/*******************************************************************************
* Function Name: Isr_Timer
********************************************************************************/
void Isr_Timer(void)
{
    /* Clear the TCPWM peripheral interrupt */
    Cy_TCPWM_ClearInterrupt(Timer_HW, Timer_CNT_NUM, CY_TCPWM_INT_ON_TC );

    /* Clear the CM4 NVIC pending interrupt for TCPWM */
    NVIC_ClearPendingIRQ(Isr_Timer_cfg.intrSrc);
       LEDupdateFlag = true;
}
/* [] END OF FILE */
```

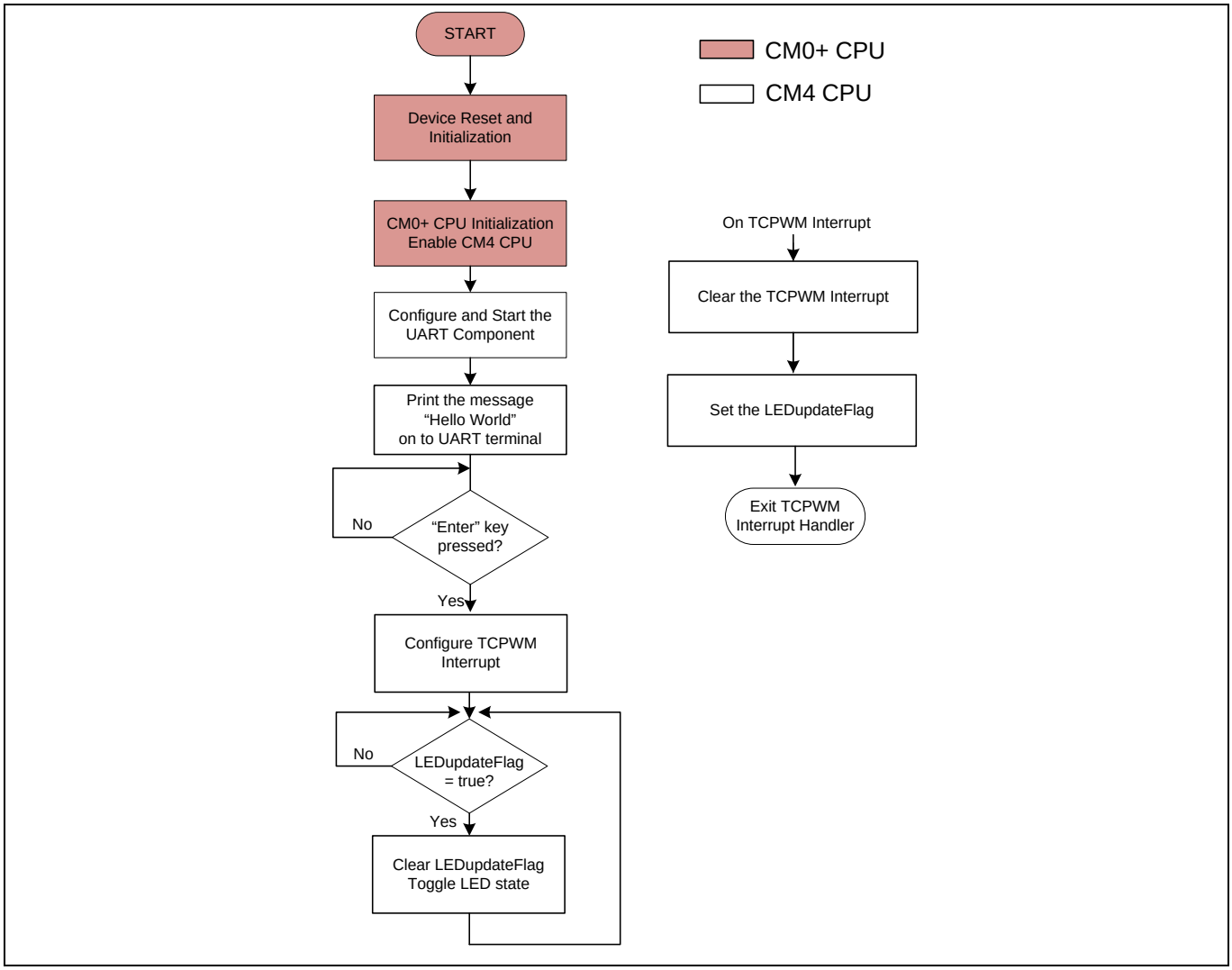

Figure 26 Firmware Flowchart 固件流程图

这样就完成了代码示例中固件工作原理的摘要。随意浏览源文件以更深入了解。

4.7 第 5 部分: 构建项目并对设备进行编程

本节介绍如何对 PSoC™ 6 MCU 器件进行编程。如果您使用带内置编程器的开发套件 (例如 CY8CKIT-062-WiFi-BT Pioneer 套件)，请使用 USB 线将电路板连接到计算机。如果您使用自己的硬件进行开发，则可能需要硬件编程/调试;例如，**CY8CKIT-002 MiniProg3**。

如果您从头开始工作并遇到错误，请重新访问先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 可以一次调试一个 CPU。

   a) 在 PSoC™ Creator 中，选择 Debug > Select Debug Target，如 **Figure 27** 所示。

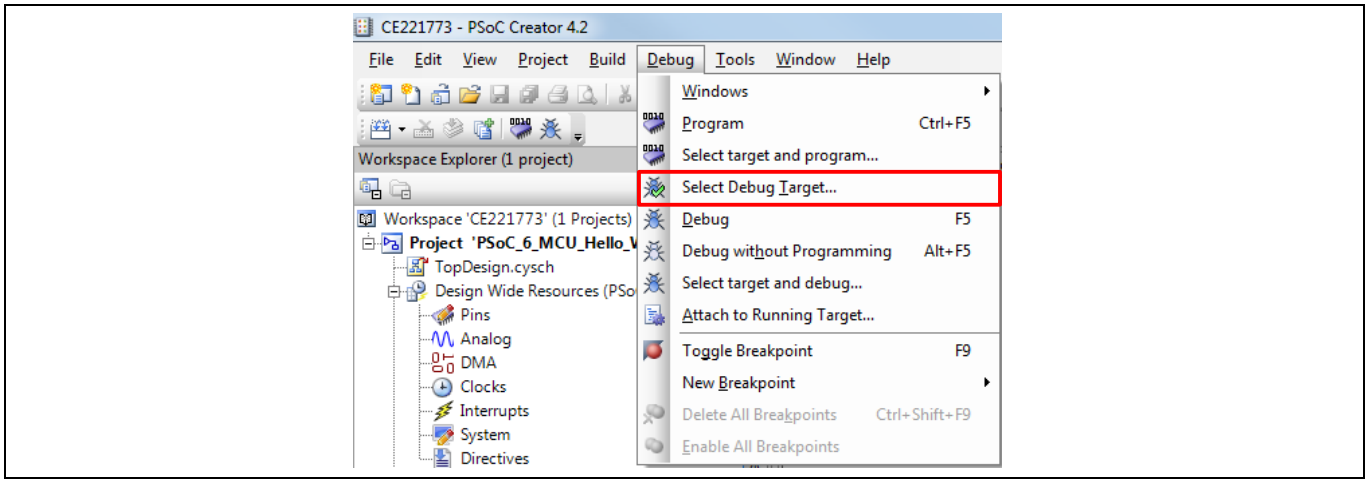

Figure 27　选择调试目标

b) 连接到电路板。
在 Select Debug Target 对话框中，选择 CM4 目标，然后单击 OK / Connect，如 **Figure 28** 所示。

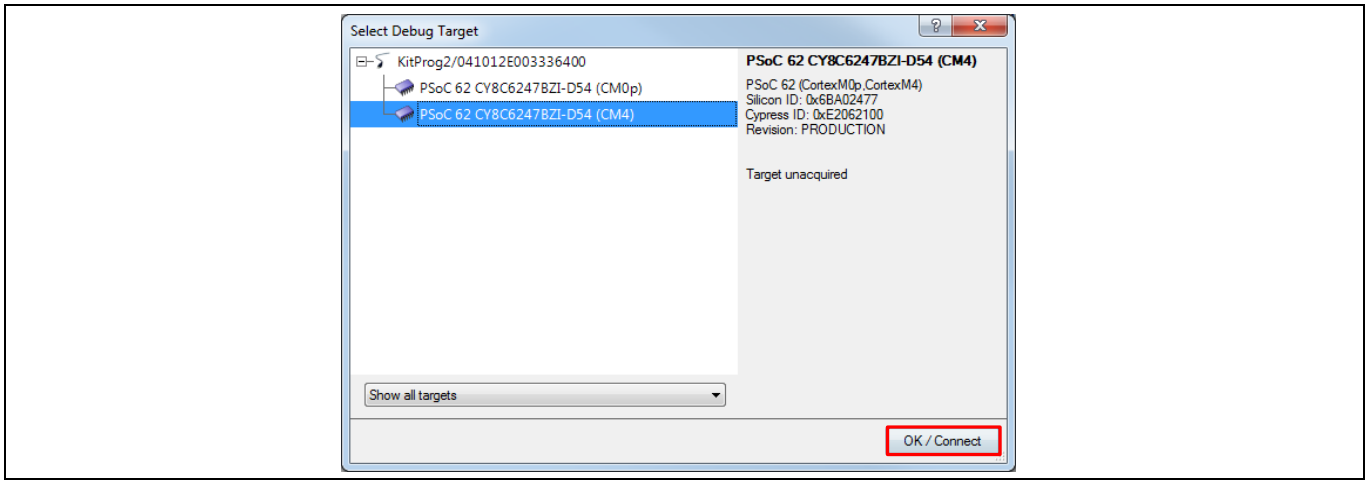

Figure 28　连接到设备

Note: *对于电路板编程，您可以选择任一目标。CPU 共享相同的内存空间。编程任一 CPU 会对两个 CPU 都进行编程。但是，如果您正在调试，这个选择很重要。调试器只能看到你连接的 CPU。这些指令不使用调试器。*

7. 编程电路板

选择 **Debug** > **Program** 对项目的设备进行编程，如 **Figure 29** 所示。

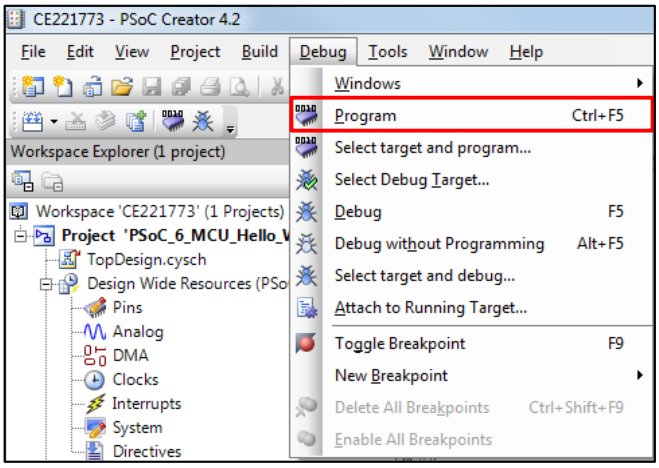

Figure 29　对设备进行编程

您可以查看在 PSoC™ Creator 中窗口左下角的编程状态，如 **Figure 30** 所示。

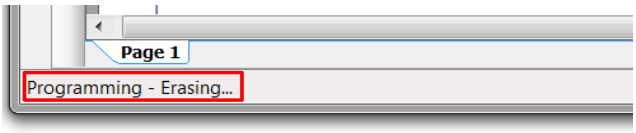

Figure 30　编程状态

Note: ***Debug*** *>* ***Debug*** *命令也可以对电路板进行编程。如果需要生成或重建任何代码，则在发出* ***Program*** *或* ***Debug*** *命令时会自动执行此操作。您也可以在不编程电路板的情况下进行调试。但是，这些指令不使用调试器。*

Note: *套件上的 KitProg2 固件可能需要更新。有关更新固件的逐步说明，请参阅相应的套件用户指南。*

4.8　第 6 部分: 测试您的设计

本节介绍如何测试您的设计。

请按照以下步骤观察设计的输出。请注意，以下步骤使用 Tera Term 作为 UART 终端仿真器来查看结果。您可以使用您选择的任何终端查看输出。

1. 选择串口

   启动 Tera Term 并选择 KitProg2 USB-UART COM 端口，如 **Figure 31** 所示。

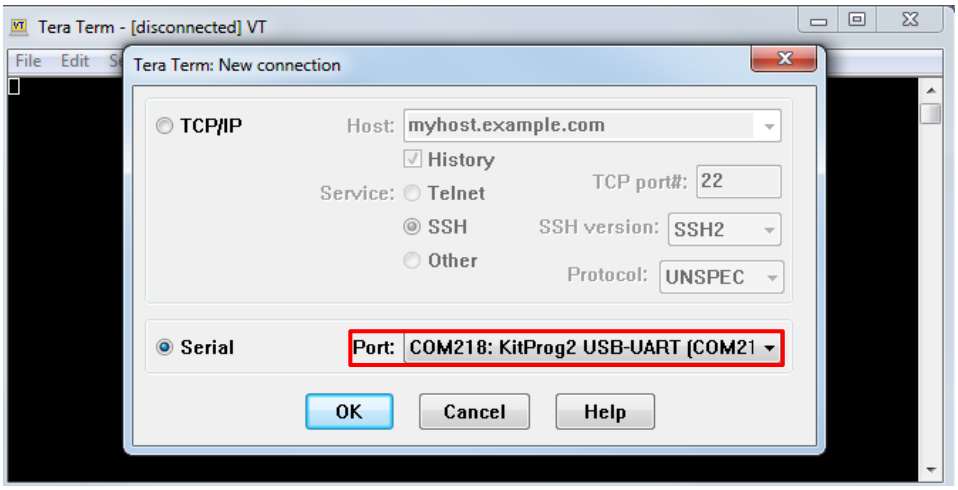

Figure 31　选择 Tera Term 中的 KitProg2 USB-UART COM 端口

2. 设置波特率

在 **Setup** > **Serial port** 下将波特率设置为 115200，如 **Figure 32** 所示。

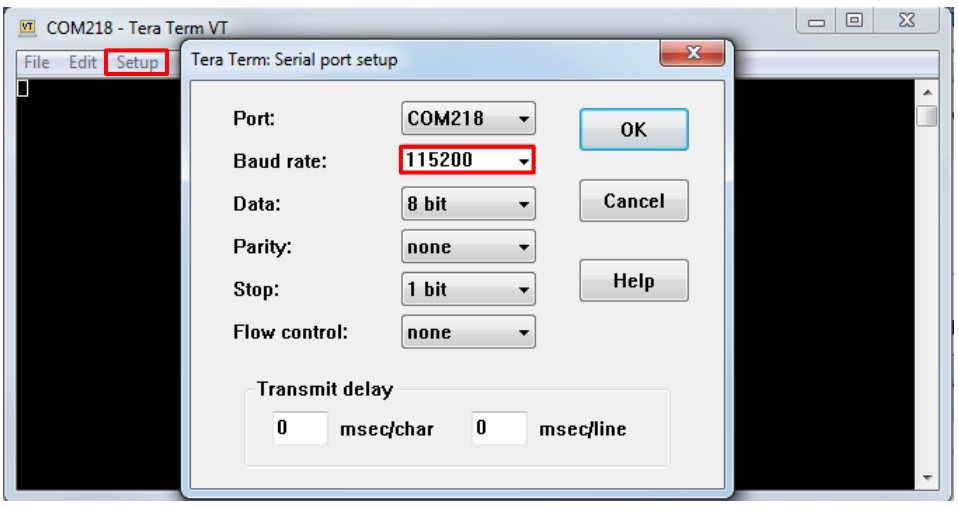

Figure 32　在 Tera Term 中配置波特率

3. 重置设备

按下 Pioneer Kit 上的复位开关 (**SW1**)。终端上显示 **Figure 33** 所示消息。

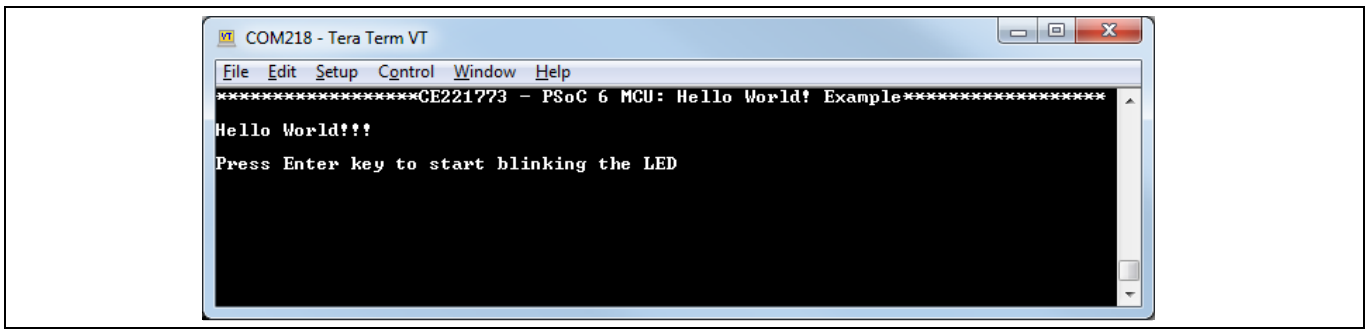

Figure 33 从 CM4 CPU 打印的 UART 消息

4. 启用 LED 闪烁功能

按 **Enter** 键开始闪烁 LED。当 LED 开始闪烁时，UART 端子上将显示以下信息，如 **Figure 34** 所示。

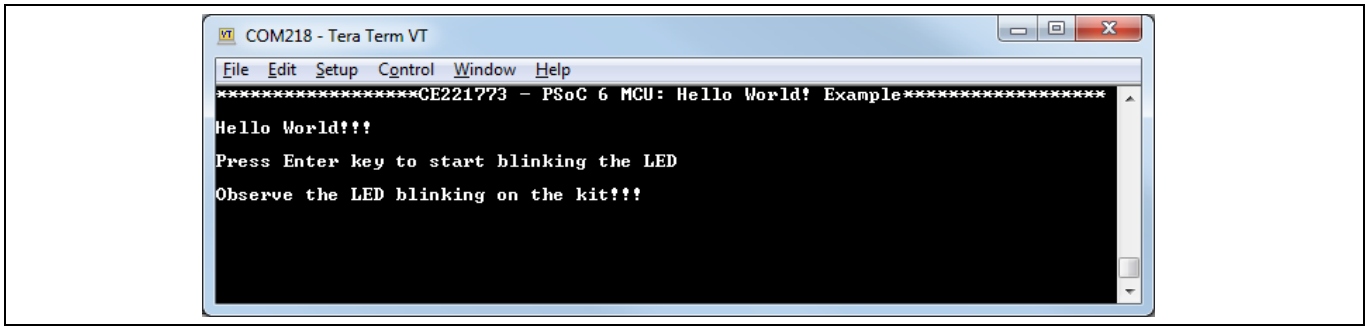

Figure 34 来自 CM4 CPU 的 UART 信息

5 总结

本应用笔记探讨了 PSoC™ 6 MCU 器件架构和相关的开发工具。PSoC™ 6 MCU 是一款真正可编程的嵌入式片上系统，在单芯片上具有可配置的模拟和数字外设功能，存储器和双 CPU 系统。整合的功能和低功耗模式使 PSoC™ 6 MCU 成为智能家居、物联网网关和其它相关应用的理想选择。

相关应用笔记和代码示例

有关 PSoC™ 6 MCU 代码示例的完整和更新列表，请访问我们的代码示例网页。有关 **PSoC™ 6 MCU** 相关文档的更多信息，请访问我们的 **PSoC™ 6 MCU** 产品网页。

Table 2 列出了系统级和一般应用笔记，建议用于学习 PSoC™ 6 MCU 和 PSoC™ Creator 的后续步骤。

Table 2　　通用和系统级应用笔记

文档	文档名称
AN228571	Getting Started with PSoC™ 6 MCU on ModusToolbox
AN210781	Getting Started with PSoC™ 6 MCU with Bluetooth® Low Energy (Bluetooth® LE) Connectivity on PSoC™ Creator
AN218241	PSoC™ 6 MCU Hardware Design Considerations
AN219434	PSoC™ 6 MCU Importing Generated Code into an IDE
AN219528	PSoC™ 6 MCU Low-Power Modes and Power Reduction Techniques

Table 3 列出了特定外设和应用的应用笔记 (AN) 和代码示例 (CE)。

Table 3　　与 PSoC™ 6 MCU 功能相关的文档

文档	文档名称
系统资源, CPU, 和中断	
AN215656	PSoC™ 6 MCU Dual-CPU System Design
AN217666	PSoC™ 6 MCU Interrupts
CE221773	PSoC™ 6 MCU Hello World Example
CE216795	PSoC™ 6 MCU Dual-Core Basics
CE216825	PSoC™ 6 MCU Real-Time Clock Basics
CE218129	PSoC™ 6 MCU Wake up from Hibernate Using Low-Power Comparator
CE218541	PSoC™ 6 MCU Fault-Handling Basics
CE218542	PSoC™ 6 Custom Tick Timer Using RTC Alarm Interrupt
CE218552	PSoC™ 6 MCU UART to Memory Buffer Using DMA
CE218964	PSoC™ 6 MCU RTC Daily Alarm
CE219339	PSoC™ 6 MCU MCWDT and RTC Interrupts (Dual Core)
CE219521	PSoC™ 6 MCU GPIO Interrupt
CE219881	PSoC™ 6 MCU Switching Power Modes
CE220060	PSoC™ 6 MCU Watchdog Timer
CE220061	PSoC™ 6 MCU Multi-Counter Watchdog Interrupts
CE220120	PSoC™ 6 MCU Blocking Mode Flash Write
CE220169	PSoC™ 6 MCU Periodic Interrupt Using TCPWM
GPIO	
CE219490	PSoC™ 6 Breathing LED Using SMART IO
CE219506	PSoC™ 6 Clock Buffer Using SMART IO
CE220263	PSoC™ 6 MCU GPIO Pins Example

相关应用笔记和代码示例

文档	文档名称
CAPSENSE™	
AN92239	Proximity Sensing with CAPSENSE™
AN85951	PSoC™ 4 and PSoC™ 6 MCU CAPSENSE™ Design Guide
Bootloader	
AN213924	PSoC™ 6 MCU Bootloader Software Development Kit (SDK) Guide
CE213903	PSoC™ 6 MCU Basic Bootloaders
通信	
CE220541	PSoC™ 6 MCU SCB EzI2C
音频	
CE218636	PSoC™ 6 MCU Inter-IC Sound (I2S) Example
CE219431	PSoC™ 6 MCU PDM-to-PCM Example
RTOS	
CE217911	PSoC™ 6 MCU FreeRTOS™ Example Project
安全	
CE220465	PSoC™ 6 MCU Cryptography – AES Demonstration
CE220511	PSoC™ 6 MCU Cryptography – SHA Demonstration

术语表

本节列出了使用 PSoC™系列器件时可能遇到的最常用术语。

- **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 Bluetooth® LE 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。
- **KitProg:** KitProg 是一个板载编程器/调试器，具有 USB-I2C 和 USB-UART 桥接功能。KitProg 集成在大多数 PSoC™开发套件中。
- **MiniProg3** / **MiniProg4**: 用于开发的编程硬件，用于编程定制板上的 PSoC™器件或不支持内置编程器的 PSoC™开发套件。
- **个性:** 个性表示功能的资源的可配置性。例如，SCB 资源可以配置为 UART，SPI 或 I2C 个性。
- **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。
- **ModusToolbox**: 基于 Eclipse 的嵌入式设计平台，面向物联网设计人员，提供简单、连贯且熟悉的设计体验，并结合业界部署最多的 Wi-Fi 和蓝牙技术，功耗最低、功能最灵活的 MCU，以及业界最佳的传感功能。
- **PSoC™ Creator**: PSoC™ 3，PSoC™ 4，PSoC™ 5LP，PSoC™ 6 MCU 和 PSoC™ 6 Bluetooth® LE 集成设计环境 (IDE) 软件，可安装在 PC 上，允许 PSoC™系统的并发硬件和固件设计，或硬件设计，然后导出到其他流行的 IDE。
- 外设驱动库: 外设驱动程序库 (PDL) 简化了 PSoC™ 6 MCU 架构的软件开发。PDL 减少了理解寄存器使用和位结构的需要，从而简化了大量可用外设的软件开发。
- **PSoC™ 编程器**: 用于编程 PSoC™器件的灵活集成编程应用程序。PSoC™ Programmer 与 PSoC™ Creator 集成，可对 PSoC™ 3，PSoC™ 4，PRoC，PSoC™ 5LP，PSoC™ 6 MCU 和 PSoC™ 6 Bluetooth® LE 设计进行编程。
- **WICED**: 英飞凌的 WICED (嵌入式设备的无线互联网连接) 是一个功能齐全的平台，拥有成熟的软件开发套件 (SDK) 和来自合作伙伴的交钥匙硬件解决方案，可以在系统设计中轻松实现 Wi-Fi 和蓝牙连接。

文档修订记录

版本	提交日期	变更说明
**	2019-04-12	翻译自 002-21774 Rev *D
*A	2020-04-20	翻译自 002-21774 Rev *E
*B	2022-05-13	翻译自 002-21774 Rev *F，更新至 Infineon 模板。

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

Edition 2022-05-13
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2022 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Go to www.infineon.com/support

Document reference
002-26896 Rev. *B

重要提示

本文档所提供的任何信息**绝不应当**被视为针对任何条件或者品质而做出的保证（质量保证）。英飞凌对于本文档中所提及的任何事例、提示或者任何特定数值及/或任何关于产品应用方面的信息均在此明确声明其不承担任何保证或者责任，包括但不限于其不侵犯任何第三方知识产权的保证均在此排除。

此外，本文档所提供的任何信息均取决于客户履行本文档所载明的义务和客户遵守适用于客户产品以及与客户对于英飞凌产品的应用所相关的任何法律要求、规范和标准。
本文档所含的数据仅供经过专业技术培训的人员使用。客户自身的技术部门有义务对于产品是否适宜于其预期的应用和针对该等应用而言本文档中所提供的信息是否充分自行予以评估。

如需产品、技术、交付条款和条件以及价格等进一步信息，请向离您最近的英飞凌科技办公室接洽(**www.infineon.com**)。

警告事项

由于技术所需产品可能含有危险物质。如需了解该等物质的类型，请向离您最近的英飞凌科技办公室接洽。

除非由经英飞凌科技授权代表签署的书面文件中做出另行明确批准的情况外，英飞凌科技的产品不应当被用于任何一项一旦产品失效或者产品使用的后果可被合理地预料到可能导致人身伤害的任何应用领域。