PSoC™ 6 MCU 导入生成代码到 IDE

相关部件系列:

PSoC™ 6 MCU

相关代码示例:

CE212736

相关应用笔记:

AN210781

关于本文档

范围及目的

AN219434 描述了如何将 PSoC™ Creator 为 PSoC™ 6 MCU 架构生成的代码导入首选的集成开发环境。有了这些知识，您就可以将自动生成的代码的优势与首选 IDE 相结合。本文档解释了高级选项，并提供了每个选项的详细说明。

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。

目录

相关部件系列: **1**
相关代码示例: **1**
相关应用笔记: **1**
关于本文档 **1**
目录 **1**
1 简介 **3**
2 集成产生的代码 **4**
2.1 生成代码的优点 5
2.2 生成代码的限制 5
2.3 工具兼容性 5
3 导出和导入生成的代码 **6**
3.1 启用目标 IDE 的导出 7
3.2 生成导出包 7
3.3 导入包 7
3.4 支持迭代开发 8

目录

3.5 导出/导入审查 8
4 手动导入生成代码 10
4.1 创建项目文件 10
4.2 查找和识别源文件 10
4.2.1 用户文件 11
4.2.2 设计文件 12
4.2.3 库文件 13
4.3 添加文件到项目 14
4.3.1 从何处获取 PDL 用户文件 14
4.3.2 从何处获取 PDL 库文件 15
4.4 支持迭代开发 15
4.5 手动导入审核 15
5 手动导入生成的代码 - 示例 16
5.1 完成设计并生成应用文件 16
5.2 设置 IDE 项目文件 17
5.3 添加文件到项目中 18
5.3.1 添加固件文件 19
5.3.2 添加用户文件 19
5.3.3 添加设计文件 20
5.3.4 添加 PDL 库文件 21
5.3.5 添加 Bluetooth® LE 库文件 22
5.4 选择包含路径 23
5.4.1 设置用户文件路径 23
5.4.2 设置设计文件的路径 23
5.4.3 设置外设文件夹的路径 (驱动程序文件) 24
5.4.4 设置中间件文件夹路径(Bluetooth® LE 堆栈) 24
5.4.5 设置其它需要的路径 25
5.5 在 IDE 中构建项目 26
5.6 示例评审 26
6 总结 27
7 相关文档 28
8 附录 A. 配置 IDE 项目文件 29
9 附录 B. 将生成代码用于学习 30
文档修订记录 31

1 简介

许多客户使用 PSoC™ Creator 工具作为硬件设计平台，但在不同的环境中编写应用软件。这可能有几个原因。您所在的机构可能拥有一套用于开发产品的既定工具。您可能希望重用另一个 IDE 中构建的遗留代码。或者您可能有一个更喜欢的开发环境。本应用笔记介绍如何在此类 IDE 中使用 PSoC™ Creator 为 PSoC™ 6 MCU 生成的代码。

PSoC™ 6 MCU 是基于双 CPU Arm® Cortex®-M4 (CM4) 和 Cortex®-M0+ (CM0+) 可编程片上系统。

为了支持 PSoC™ 6 MCU 器件的固件开发，赛普拉斯提供了两种不同的开发环境：一种基于 PSoC™ Creator，另一种基于 ModusToolbox v2.x 软件。**Table 1** 总结了每个工作流程的关键功能。参见 **ModusToolbox User Guide** 中 ModusToolbox 软件概览一节快速了解 ModusToolbox v2.x 提供的功能以及工作模式。

Table 1 PSoC™ Creator 与 ModusToolbox 软件比较

特性	PSoC™ Creator	Eclipse IDE for ModusToolbox
IDE Framework	专用，不可扩展	基于 Eclipse，可扩展
驱动库	外设驱动库 (PDL)	GitHub 上的 psoc6pdl 和 psoc6hal 库
代码生成	组件 (包括 Bluetooth® LE 和 CAPSENSE™)	配置器: 器件, CAPSENSE™, 蓝牙, QSPI, SmartIO, SegLCD, USBDev
主机操作系统	Windows	Windows, macOS, Linux
中间件	Bluetooth® LE, dfu, em_eeprom, emWin, usb_dev, FreeRTOS	用于多个生态系统的大量的库集合。参见 GitHub 上的 **PSoC™ 6 Middleware**。
后续处理工具	cymcuelftool (Windows, macOS, Linux)	大多数用例不需要，仍然支持 cymcuelftool。

本应用笔记讨论了 PSoC™ Creator 的环境和工作流程。**AN225588** (ModusToolbox 软件与第三方 IDE 结合使用) 讨论了该环境和工作流程。

外设驱动程序库 (PDL) 是用于 PSoC™ 6 MCU 器件的软件开发套件 (SDK)。您的固件使用 PDL API 函数调用来配置、初始化、启用和使用外设驱动程序。PDL 还包括对中间件的支持，例如蓝牙低功耗 (Bluetooth® LE) 和实时操作系统 (RTOS)。根据设计，PDL 对 IDE 没有特殊要求。

但是，PSoC™ Creator 与 PDL 完全集成。作为设计环境，PSoC™ Creator 可帮助您使用友好的用户界面配置时钟、中断、引脚分配和驱动程序。要自定义驱动程序，请将组件添加到与外设对应的 PSoC™ Creator 设计中，而不是编写配置代码、修改组件中的选项。然后，PSoC™ Creator 生成所有配置代码，以根据设计设置时钟、中断、引脚和自定义驱动程序。在针对 PSoC™ 6 MCU 器件时，PSoC™ Creator 使用 PDL 生成代码。如果使用 PDL 而不使用 PSoC™ Creator，则必须为设计编写所有配置代码。

生成的代码是一种宝贵的资源，因为它可以处理为一个设计设置固件所涉及的大量复杂问题。本应用笔记介绍了如何在任何 IDE 中使用该生成的代码。有多种方法可以做到这一点。本应用笔记探讨并解释了您的选择。它还通过一个示例向您展示如何完成它。读完本应用笔记后，您将了解自己的选择，以及如何将生成的代码集成到首选 IDE 中。

即使现有情况让你无法直接使用生成的代码，也不是完全没有机会。此应用笔记介绍了如何使用生成的代码学习 PDL。

要了解有关 PDL 或 PSoC™ Creator 的更多信息，请参阅“**相关文档**”部分以获取某些可用资源的链接。

2 集成产生的代码

在较高的层次上，有两种方法可以将生成的代码集成到 IDE 的项目文件中。这两条路径是：

- 导出代码并将该代码导入 IDE。
- 手动将生成的源文件添加到 IDE 的项目中。

Figure 1 中的流程图显示了您为每个路径执行的任务。本应用笔记介绍了流程图中提到的一次性任务。根据您的情况，一条或另一条路径可能更适合您。

导出路径仅适用于支持的 IDE。有关此路径的详细信息，请参阅**导出和导入生成的代码**。支持的 IDE 包括：

- IAR Embedded Workbench
- Keil µVision
- 基于 Eclipse IDE

手动导入代码适用于任何 IDE，包括支持的 IDE。对于支持的 IDE，可以使用其他资源来简化手动方法，包括完全配置的项目文件，IDE 特定的启动代码，闪存配置文件和链接脚本。有关此路径的详细信息，请参阅**手动导入生成代码**。

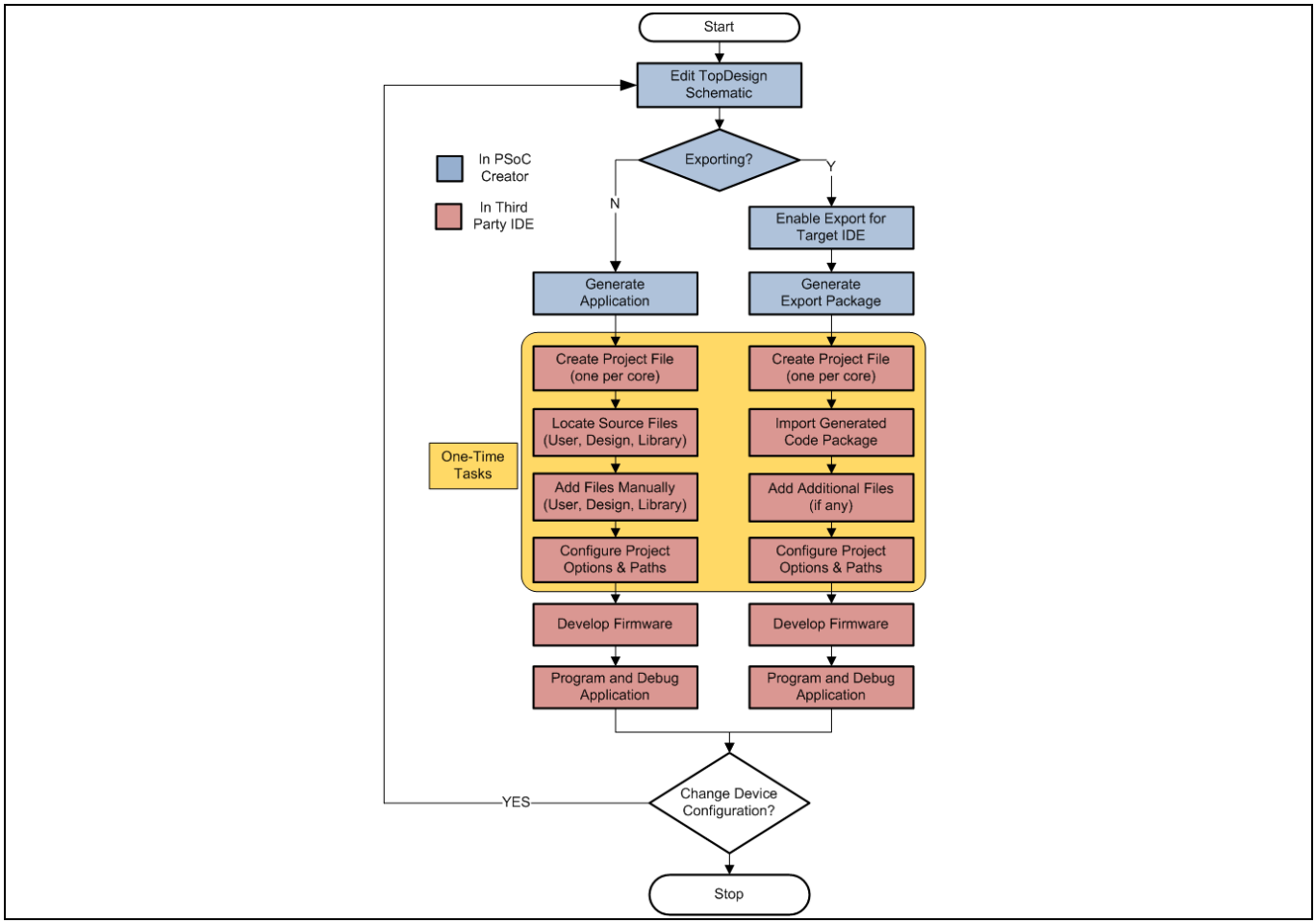


Figure 1 集成产生的代码

集成产生的代码

2.1 生成代码的优点

如果使用 PSoC™ Creator 设计和配置系统，则可以获得生成代码的显著优势。将为您创建所有必需的配置代码。这段代码可能非常复杂。例如，cyfitter_cfg.c 中生成的代码初始化所有时钟、设置每个时钟源和分频器，并启用设计中的所有时钟。它使 PDL API 函数调用这样做。此代码基于 PSoC™ Creator 设计中的时钟树配置。

Bluetooth® LE 组件是另一个很好的例子。配置 Bluetooth® LE 组件的代码通常是数百行，声明了十几个配置结构，并定义了超过 100 个字段的值。PSoC™ Creator 根据设计中的 Bluetooth® LE 组件配置自动生成所有这些代码

除配置代码外，PSoC™ Creator 还为每个组件生成简化的函数 API。组件 API 构建于并使用 PDL API。这意味着您可以使用组件 API 而不是直接调用 PDL 函数。您可以混合和匹配这些 API。它们是一致且兼容的。

例如，要使用 PDL API 初始化、启用和启动 PWM，您的代码可能如下所示 (忽略错误)：

```
Cy_TCPMW_PWM_Init (TCPWM1, counterNumber, &PWM_config);
Cy_TCPWM_Enable_Multiple (TCPWM1, whichCounters);
Cy_TCPWM_TriggerStart (TCPWM1, whichCounters);
```

相比之下，使用组件 API，您的代码将如下所示 (对于名为 MyPWM 的 TCPWM 组件)：

```
MyPWM_Start();
```

代码生成器执行适当的函数调用序列以启用 PWM。它还根据 PSoC™ Creator 中的设计和配置提供所有与硬件相关的参数。如果检查 `MyPWM_Start()`函数的生成代码，则会找到为您执行的所有三个 PDL API 调用，并以正确的顺序调用。

生成的代码主要在 C 源文件和头文件中，但可能包括汇编程序文件和已编译的二进制文件。如果您正确地将这些生成的文件添加到 IDE 项目中，如本应用笔记所述，则完全支持迭代开发。在 PSoC™ Creator 中修改设计并重新生成应用程序时，IDE 会识别文件已更改，无需额外的工作。

2.2 生成代码的限制

PSoC™ Creator 生成与 C99 标准兼容的代码。您使用的编译器必须支持该标准。此外，编译器的实现细节也稍有不同。特定编译器总是有可能以非标准方式处理某些语法，但这是不寻常的

最大的限制是在大多数情况下，您无法修改生成的代码。如果修改生成的源文件中的代码，则重新生成代码时更改将丢失。您对生成的代码所做的更改不会上行迁移到 PSoC™ Creator 设计中。如果在开发周期中您发现需要修改设计，请在 PSoC™ Creator 中进行更改，然后重新生成应用程序。

但是，PSoC™ Creator 生成的某些文件被视为用户文件。您可以修改任何用户文件而不会丢失更改。请参阅**用户文件**。

2.3 工具兼容性

PSoC™ Creator 和 PSoC™ Programmer 是为 Windows 操作系统构建的专用工具。使用其他提供商的工具时，可能会遇到兼容性问题。例如，PSoC™ Creator 将专有信息添加到最终的 hex 文件中，而 PSoC™ Programmer 则需要此信息。其他 IDE 不会生成此信息，导致 PSoC™ Programmer 不能使用此类 hex 文件。

ModusToolbox 软件和更新的 Cypress Programmer 没有这些问题。

3 导出和导入生成的代码

导出/导入路径仅适用于支持的 IDE。对于支持的 IDE，PSoC™ Creator 提供导出包。**Table 2** 列出了导出包以及每个包支持的 IDE。

Table 2 可用的导出包

导出包	支持的 IDE	主要文件	路径
CMSIS 包	Keil µVision v5 或更高版本 IAR Embedded Workbench v8.或更高版本 能导入 CMSIS Pack 的基于 Eclipse 的 IDE。	.pack file	*Export/Pack*
项目间连接文件(IPCF)	IAR Embedded Workbench	.ipcf file (每 CPU 一个)	*Export*
Makefile	GNU 命令行工具	makefile	*<project>.cydsn*

所有支持的 IDE 的导出/导入过程都是相同的。开发设计后，请执行以下步骤：

- 在 PSoC™ Creator 中:
  - 启用导出包的创建
  - 编译代码 (这会生成导出包)
- 在第三方 IDE 中：
  - 创建一个空项目 (多 CPU 设备每个 CPU 一个)
  - 将包导入空项目
  - 如有必要，请添加导出包中未包含的其他文件
  - 配置项目选项

导出/导入过程的详细信息在支持的 IDE 之间差异很大。本节将帮助您了解该过程的工作原理。PSoC™ Creator 帮助和 **PSoC™ Creator User Guide** 中介绍了确切的步骤和精确的详细信息。本应用笔记不再重复这些信息。您必须使用该文档成功导出包并将其导入 IDE。

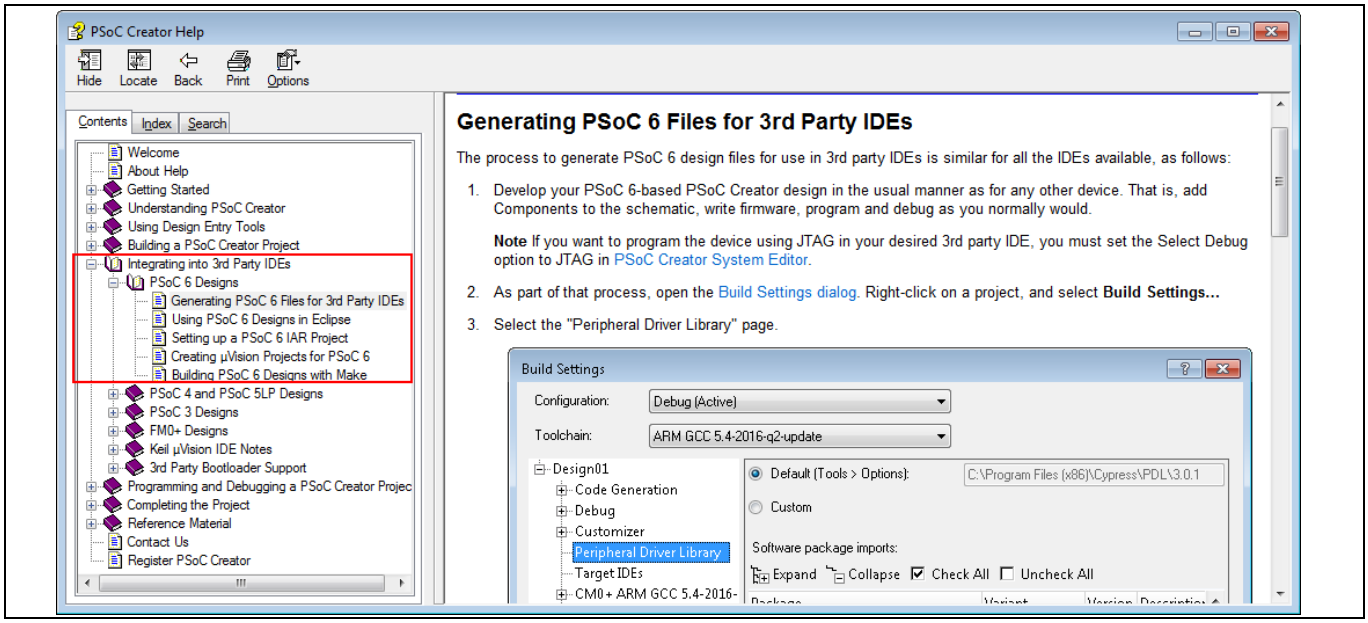


Figure 2 用于集成到第三方 IDE 的 PSoC™ Creator 帮助主题

导出和导入生成的代码

3.1 启用目标 IDE 的导出

在 PSoC™ Creator 中，使用 **Project** > **Build Setting** 菜单命令打开项目设置。然后，打开**目标 IDE** 面板。为要创建的导出包选择“**Generate**”。**Figure 3** 显示了启用的 CMSIS Pack 选项。**CMSIS Pack** 是一个与 IDE 无关的软件组件交付包。

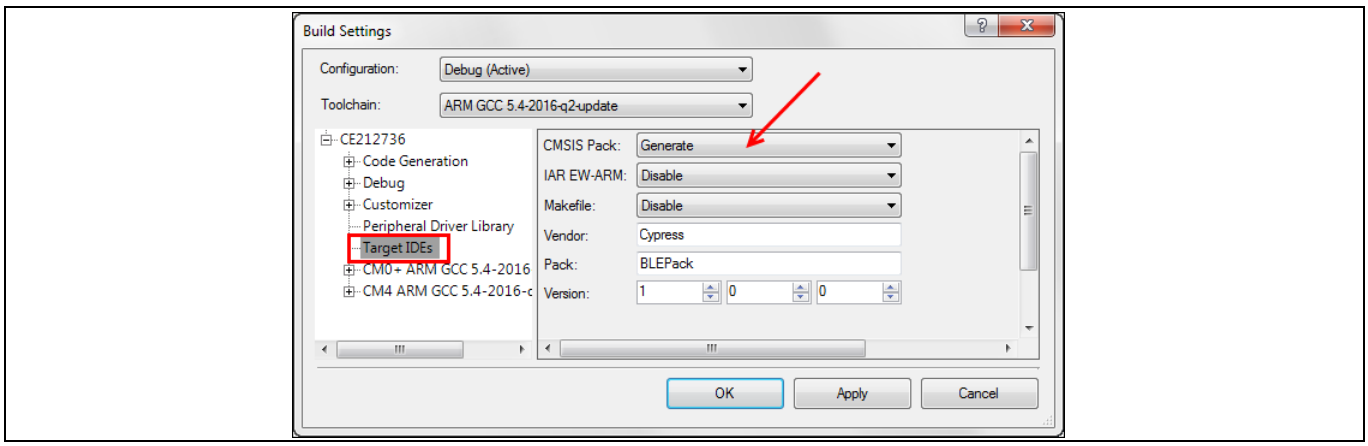


Figure 3 启用 CMSIS Pack 导出包

作为导出过程如何变化的示例，图中显示的 **Vendor**，**Pack** 和 **Version** 信息仅对 CMSIS Pack 是必需的。**IAR EW-Arm®**有不同的选择。

有关详细信息，请参见 PSoC™ Creator 文档。

3.2 生成导出包

启用包的创建后，编译您的代码。使用 **PSoC Creator Build> Build <project>**命令。请注意，**Build** > **Generate Application** 命令不会生成导出包。

Export 文件夹的内容因包而异。该包是一组文件。PSoC™ Creator 将每个包放在 **Table 2** 中列出的位置。导入包时，导航到该位置以查找所需的文件。

3.3 导入包

根据 IDE 和包，导入步骤变化很大。本节介绍导入包时必须执行的常规任务。有关准确的步骤和详细信息，请参阅 PSoC™ Creator 文档。每个支持的 IDE 都有一个帮助主题，如 **Figure 2** 所示。

通常，您执行以下任务：

- 创建一个空项目文件 (多 CPU 设备每个 CPU 一个文件)。
- 将生成的代码导入每个项目文件。**Figure 4** 显示了每个包的位置。
  - 对于 CMSIS 包，请在计算机上安装该包，然后导入该包。
  - 对于 IAR 工具，使用*.ipcf* 文件建立项目连接。
  - 对于 GNU 工具，请使用生成的 makefile。

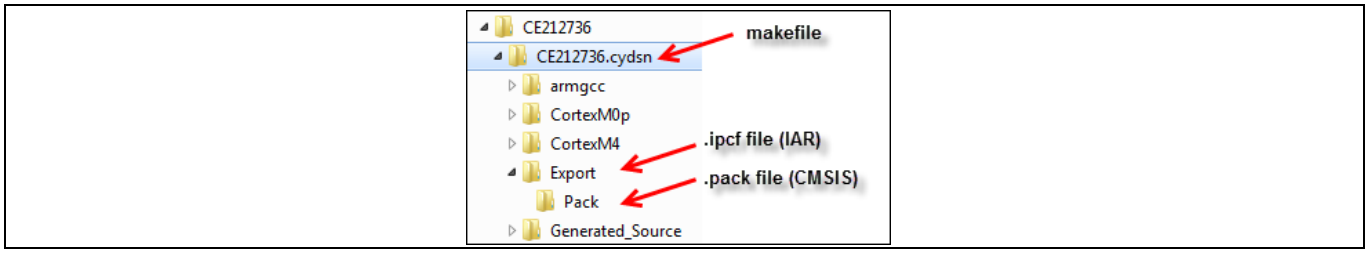


Figure 4　在何处查找导出包

- 在某些情况下，请为每个项目文件添加其他文件：
  - 对于 CMSIS Pack，您可以从模板创建特定 CPU 的 *main.c* 和链接器文件。
  - 对于 IAR 工具，这些文件会自动包含在项目中。
- 设置项目选项，其中可能包括编译器、链接器和调试器设置。

不同的包需要您设置不同的项目选项。例如，IAR 工具的 IPCF 连接要求您修改 CM4 项目的输出文件夹名称，这样它们不会与 CM0+项目冲突。对于 CMSIS Pack，还可以指定头文件的包含路径，提供链接器命令文件的路径，等等。PSoC™ Creator 文档提供了所有详细信息。在导出和导入生成的代码之前，请确保您熟悉该文档。有关项目选项的更多信息，请参阅**设置 IDE 项目文件**。

项目中的几个文件是特定 CPU 或 IDE 的，例如链接器文件、启动代码、makefile 等。每个 CPU 和 IDE 的正确文件都在包中。导入包时，包含了正确的文件。

Note:　*许多特定 CPU 文件是用户可编辑的文件，最初由 PSoC™ Creator 生成。您可以修改这些文件，并且在重建 PSoC™ Creator 项目时不会丢失更改。有关更多信息，请参阅**用户文件**。*

3.4　支持迭代开发

导入生成的代码后，您可能会有更改设计的需要。在 PSoC™ Creator 项目中进行必要的更改，然后重建项目。不要修改 *Generated_Source* 文件夹中的文件。重新生成代码时，任何更改都将丢失。

在 PSoC™ Creator 中重建代码时，*Export* 文件夹中会显示更新的包。代码生成过程可能具有已修改的文件，添加的新文件或已删除的文件。

对于 CMSIS Pack 导出，请重新安装 Pack。在 *Export* 文件夹中找到包文件，然后双击以进行安装。解压缩对话框包含已安装 Pack 的警报。代替它。IDE 会自动识别更改，包括添加或删除的文件。在 IDE 中重建代码以更新可执行文件。

IAR 项目间连接文件指向 PSoC™ Creator *Generated_Source* 文件夹。IDE 会自动识别更改，包括添加或删除的文件。无需重新连接到.*ipcf* 文件。只需在 IDE 中重建代码即可更新可执行文件。

对于 makefile，请确保构建过程使用 *Export* 文件夹中的文件。如果是，那么用新代码对可执行文件进行简单的清理和构建。

3.5　导出/导入审查

PSoC™ Creator 文档描述了所有详细信息。导入过程从 IDE 中的新空项目开始。

对于 CMSIS Pack，IDE 的项目设置涉及多个步骤。由于 Pack 可以导入到各种 IDE 中，并且每个 IDE 在处理构建选项方面都是独一无二的，因此您需要设置多个选项。这是一次性任务。代码中的任何更改 (来自 PSoC™ Creator 中的设计更改) 都是透明处理的。只需重新安装包。

导出和导入生成的代码

项目间连接是 IAR 工具专有的协议。由于它是为一个 IDE 设计的，因此项目连接会自动设置大多数选项。这是一个非常友好的过程。

4 手动导入生成代码

手动导入适用于任何 IDE，包括具有导出/导入过程的受支持 IDE。对于不受支持的 IDE，手动导入是唯一的选择。

首先，您必须已生成代码。在 PSoC™ Creator 中，**Build** > **Generate Application** 命令就足够了。您不需要编译代码。要手动导入生成的代码，您：

- 在 IDE 中创建一个项目 (对于多 CPU 设备，每个 CPU 一个项目)。
- 找到并识别所需的生成文件。
- 将这些文件添加到项目中。

本节为您提供了解 PSoC™ Creator 生成的文件，在何处查找以及哪些需要添加到项目中所需的背景知识。**手动导入生成的代码 - 示例**详细介绍了该过程。

4.1 创建项目文件

本讨论假设您熟悉首选 IDE，您知道如何创建和配置项目文件，并且您已将项目配置为与多 CPU PSoC™ 6 MCU 器件上的每个 CPU 配合使用。有关必须配置的选项类型的信息，请参阅**设置 IDE 项目文件**。

您可以从一个新的空项目文件开始。但是，赛普拉斯为每个支持的 IDE 和 PSoC™ 6 MCU CPU 提供了预配置的模板项目。有 µVisionIDE、IAR IDE 和其他支持的 IDE 的 CM4 和 CM0+项目。PDL 模板项目在“**PDL v3.1 用户指南**”标题为“*使用 PDL 模板项目*”的一节中进行了完整描述。模板项目位于此处：*<PDL Install Folder>/devices/psoc6/templates*。

模板项目中的大多数构建选项都已正确设置。导入生成的代码时所需的更改将在本应用笔记中讨论。此外，模板中的包含路径是项目相关的。如果将模板项目文件移动到其他位置，则必须修改这些路径。请参阅**从何处获取 PDL 库文件**。

4.2 查找和识别源文件

PSoC™ Creator 生成三种源文件：

- 用户文件 - 您可以修改的文件
- 设计文件 - 特定于 PSoC™ Creator 设计和组件的文件
- 库文件 - 来自 PDL 的文件

以下部分详细描述了每种源文件。

PSoC™ Creator 将这些文件放在*<project>.cydsn* 文件夹中的各种文件夹中。本应用笔记将此位置简称为 cydsn 文件夹。**Figure 5** 显示了生成的源文件的文件夹树和位置。

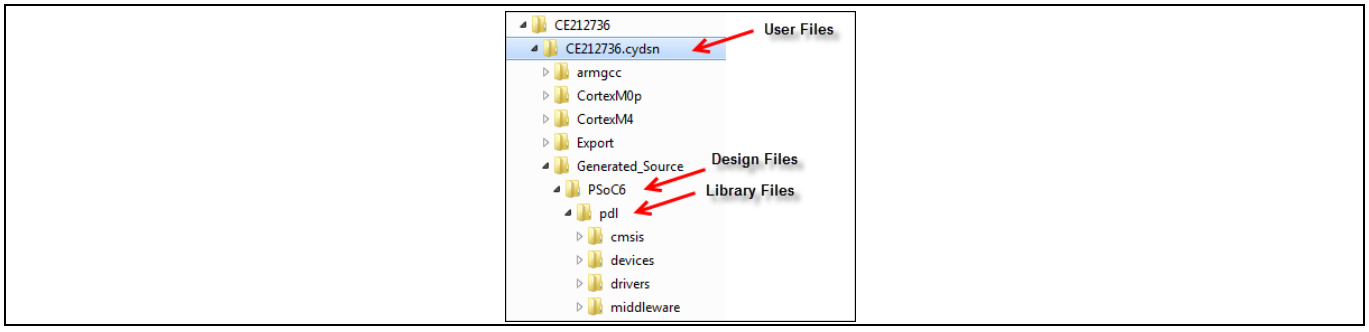


Figure 5 何处查找生成的文件

4.2.1 用户文件

用户文件是您可以根据设计要求更改的文件。PSoC™ Creator 生成文件 (可能在功能上为空)，并且永远不会更改它。这些是了解用户文件的关键概念：

- 首次创建 PSoC™ Creator 项目并生成应用程序时，将自动创建文件。
- 文件创建后 PSoC™ Creator 不会更改或替换这些文件。因此，您可以根据设计需要修改这些文件。
- 所有文件都是本地副本，因此更改不会影响其他项目或已安装的 PDL 库。
- 对于多 CPU 设备，将特定 CPU 的文件添加到 IDE 中的正确项目中;为每个 CPU 添加共享文件到项目。

您不需要每个文件;有些是特定于特定工具链的。

Table 3 列出了常见的用户文件。用户文件位于 cydsn 文件夹的顶层 (启动代码除外，如表中所示)。某些文件名因设备而异。该表使用 psoc63 系列作为示例。有些文件是针对特定 CPU 的。启动代码和链接器文件也是针对特定 IDE 的。使用适合您项目的文件。忽略其它。如果使用不受支持的 IDE，请提供与 IDE 兼容的等效文件。

根据您的设计，可能还有其他用户文件。cydsn 文件夹顶层的任何源文件都是用户文件，应该添加到一个或两个项目中，具体取决于文件是否针对特定 CPU。

Table 3 PSoC™ Creator 生成的用户文件

文件	操作	源文件	使用	备注
main_cm0p.c *main_cm4.c*	添加到项目 [1], 或提供你自己的。.	生成	需要	PSoC™ Creator 将用户文件放在 cydsn 文件夹的顶层
cyapicallbacks.h	为头文件提供路径	生成	需要	用户实现宏回调的空文件 (请参阅 PSoC™ Creator 帮助主题中的 *编写代码*)
system_psoc6.h	为头文件提供路径	从 PDL 拷贝。	需要	包含用户可定义的时钟配置宏
system_psoc6_cm4.c *system_psoc6_cm0plus.c*	添加到项目 [1]	生成	需要	根据 CPU; 系统配置代码
startup_psoc6_01_cm4.s *startup_psoc6_01_cm0plus.s* *(IAR)*	添加到项目 [1]。对于不支持的 IDE, 提供一个启动文件。	从 PDL 拷贝。	使用针对你的 IDE 的文件	根据系列、CPU 和 IDE; 启动代码位于: *cydsn/<IDE>/startup*
startup_psoc6_01_cm4.s *startup_psoc6_01_cm0plus.s* *(MDK)*				
startup_psoc6_01_cm4.S *startup_psoc6_01_cm0plus.S* *(GCC)*				
cy8c6xx7_cm4_dual.icf *cy8c6xx7_cm0plus.icf*	设定项目选项以使用指定 CPU 连接器文件。对于不支持的 IDE, 提供一个连接器文件。	从 PDL 拷贝。	使用针对你的 IDE 的文件	根据系列、CPU 和 IDE; 链接命令文件
cy8c6xx7_cm4_dual.scat *cy8c6xx7_cm0plus.scat*				
cy8c6xx7_cm4_dual.ld *cy8c6xx7_cm0plus.ld*				

[1] 将针对特定 CPU 的文件添加到对应 CPU 的项目中。

手动导入生成代码

Note: *此表中列出的所有头文件都位于同一文件夹中，因此单个路径可以访问所有文件。*

Note: *虽然默认的 cyapicallbacks.h 文件为空，但没有简单的方法可以将其从项目中删除。它包含在另一个自动生成的头文件中。如果删除包含此标头的#include 语句，则下次生成代码时该语句会重新出现。*

Note: *PDL 模板项目具有来自已安装的 PDL 位置的用户文件和路径，而不是 cydsn 文件夹中生成的文件(副本)。手动导入代码时，请将文件替换为 cydsn 文件夹中的本地副本。请参阅**从何处获取 PDL 用户文件**。*

虽然某些文件是特定 CPU 的，但有些文件是由多 CPU 设备中的每个 CPU 使用的。文件 *system_psoc6.h* 就是一个很好的例子。每个 CPU 使用相同的文件，基于用户可定义时钟配置宏来设置 CPU 时钟。手动将共享文件导入 IDE 时，必须将该文件添加到每个 CPU 的项目中。如有疑问，请右键单击 PSoC™ Creator 工作区资源管理器窗格中的文件并检查其属性，以查看哪个 CPU 工具链使用了该文件。

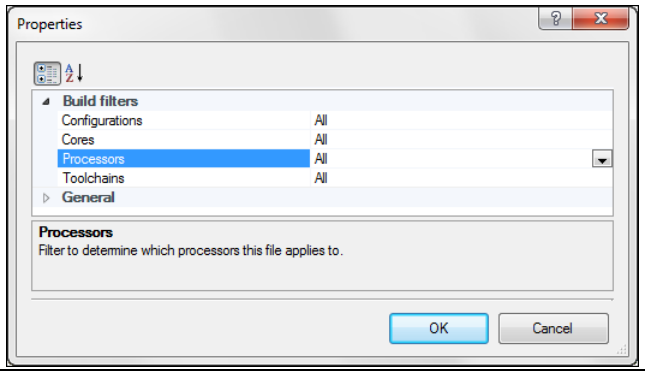


Figure 6 ***system_psoc6.h* 文件属性**

4.2.2 设计文件

PSoC™ Creator 会自动生成多个特定用于您设计的文件。在某些情况下，这些文件也特定用于受支持的 IDE。所有设计文件都在此文件夹中：*cydsn / Generated_Source / PSoC6*。

设计文件在 PSoC™ Creator 帮助的 *PSoC Creator Project > Generated Files (PSoC 6)*主题中进行了介绍。设计文件还包括设计中每个组件的源文件和头文件。如果组件基于通用数字块 (UDB)，则配置和使用组件所需的所有文件都在设计文件中。

这些是了解设计文件的关键概念：

- PSoC™ Creator 可能会在您生成应用程序时随时更改这些文件。
- 不要修改这些文件;生成应用程序时，任何更改都将丢失。
- 特定于组件的文件名为<组件名称> .h 和<组件名称> .c。
- 某些文件是可选的和/或特定于 IDE 的 (参见 **Table 4**)。

大多数设计文件都是共享文件。对于多 CPU 设备，请为每个 CPU 的项目添加共享文件。任何特定于 CPU 的文件都在文件名中标识 CPU。

Table 4 列出了设计文件以及在项目中使用文件所需的操作。

手动导入生成代码

Table 4 PSoC™ Creator 生成的设计文件

文件	操作	使用	备注
project.h	为头文件提供路径 [2]	需要	包括本表格中列出的所有头文件。
<Component Name>.c/h	添加文件到项目。 为头文件提供路径.	需要	特定组件 API 的代码。头文件包含在 *project.h* 中
cyfitter.h	为头文件提供路径.	需要	#定义用于特定于设计的地址和值
cyfitter_cfg.c/h	添加文件到项目. 为头文件提供路径.	需要	到达 main() 之前配置设备的代码
cyfitter_gpio.h	为头文件提供路径	需要	#定义用于特定设计的地址和引脚配置的值
cyfitter_sysint.h	为头文件提供路径	需要	#定义用于特定设计的中断
cyfitter_sysint_cfg.c/h	添加文件到项目。 为头文件提供路径.	需要	用于配置系统中断的代码
cymetadata.c	添加到项目.	需要	定义需要包含的所有额外内存空间。
cydevice_trm.h	为头文件提供路径	需要	定义设备配置空间中的所有地址。
cydisabledsheets.h	为头文件提供路径	需要	定义表示已禁用的原理图页面；如果没有，则为空
cydevicegnu_trm.inc *cydeviceiar_trm.inc* *cydevicerv_trm.inc*	如果需要，包含在你的汇编代码中	可选	每个 IDE; 定义汇编程序的设备配置空间中的所有地址
cycodeshareexport.ld *cycodeshareimport.ld* *cycodeshareimport.scat*	设置项目选项以在需要的情况下使用脚本	可选	每个 IDE; 用于将符号导出或导入其他应用程序的链接描述文件; 通常为空

默认的 *main_cm0p.c* 和 *main_cm4.c* 文件包含 *project.h*。*project.h* 文件又包含设计文件中所有其他必需的头文件。如果您提供自己的 *main.c* 文件，请确保包含 *project.h*。

虽然 cydisabledsheets.h 通常为空，但没有简单的方法可以将其从项目中删除。它包含在 project.h 中。如果删除#include 语句，则下次生成代码时该语句会重新出现。

4.2.3 库文件

库文件是 PDL 安装中的文件副本。PSoC™ Creator 会自动将所需文件复制到 *cydsn / Generated_Source / PSoC6 / pdl* 文件夹的子文件夹中。请参阅《PDL v3.0 用户指南》以了解 PDL。

这些是了解库文件的关键概念：

- 仅复制所需文件，而不是整个 PDL。
- 某些文件可能是汇编源文件或预编译的二进制文件。
- PSoC™ Creator 永远不会修改库文件。
- 不要修改这些文件;生成应用程序时，任何更改都将丢失。
- 大多数库文件都是共享文件。对于多 CPU 设备，请为每个 CPU 的项目添加共享文件。任何特定 CPU 的文件都在文件名中标识 CPU。

PDL 文件夹树中显示的库文件取决于您的设计。**Table 5** 列出了树中的关键位置，每个文件夹中的内容以及如何在项目中使用这些文件。

[2] 此表中列出的所有头文件都位于同一文件夹中，因此单个路径提供对所有文件的访问权限。

手动导入生成代码

Table 5 PSoC™ Creator 生成的库文件

路径/文件夹	操作	备注
pdl/cmsis/include	提供此文件夹的路径。	CMSIS 头文件
pdl/devices/psoc6/include/ip	提供此文件夹的路径。	设备上 IP 模块的头文件
pdl/devices/psoc6/include	提供此文件夹的路径。	特定于设备的头文件
pdl/drivers/peripheral	添加源文件到项目中。提供此文件夹和所有子文件夹的的路径。	设计中使用的外设驱动程序的源文件和头文件;
pdl/middleware	添加源文件到项目中。提供此文件夹路径。	项目中使用的中间件的源文件和头文件; 不要添加子文件夹的路径。

*Note: 系统库(syslib) 外设驱动程序具有汇编程序文件作为其实现的一部分。它必须与 C 源文件一起添加到项目中。**手动导入生成的代码 - 示例**向您展示如何操作。*

Note: pdl/middleware 文件夹可能有子文件夹。不要为每个子文件夹添加路径。PDL 源代码在 `#include` 语句中提供了路径。

*Note: PDL 模板项目的路径指向已安装的 PDL 位置，而不是 cydsn 文件夹中生成的文件(副本)。手动导入文件时，请重置这些路径以指向 cydsn 文件夹。请参阅**从何处获取 PDL 库文件**。*

4.3 添加文件到项目

在确定所需文件时，将它们添加到 IDE 的项目文件中。每个 IDE 都有自己的方法将文件添加到项目中。例如，IDE 可以支持拖放，或通过 IDE 的用户界面添加一批文件。

添加文件时要记住三个关键原则：

首先，直接从 PSoC™ Creator cydsn 文件夹中的位置添加各个文件。这对于支持迭代开发至关重要。如果在 PSoC™ Creator 中更改设计，则某些生成的文件将更改。通过将 IDE 项目指向 cydsn 文件夹中的文件，可以确保对文件进行的任何更改出现在项目中。

其次，一些文件是特定于 CPU 的。将特定于 CPU 的文件添加到相应的项目中。对于多 CPU 设备，请将共享文件添加到两个项目中。特定 CPU 的文件的文件名标识该 CPU。

第三，您必须添加包含路径，以便预处理器可以找到所需的头文件。好消息是大多数头文件被分组到单个文件夹中。一些开发人员更喜欢将头文件添加到项目中以便于访问该文件。即使您这样做，大多数 IDE 仍然需要为预处理器添加包含路径。

4.3.1 从何处获取 PDL 用户文件

某些 PSoC™ Creator 生成的用户文件作为默认 PDL 文件的副本启动。不要使用原始 PDL 文件。使用 cydsn 文件夹中生成的副本。

用户可以更改用户文件。如果更改原始 PDL 文件，则该更改会影响使用该文件的任何其他项目。最佳实践要求您使用 cydsn 文件夹中的副本。请参阅**用户文件**。

如果您从 PDL 模板项目开始，您应该：

- 从项目中删除用户文件 (它们是 PDL 安装的原始文件)。
- 从 cydsn 文件夹中将相同的文件添加回项目。
- 删除*/include* 文件夹的项目相对路径。它通常在 IDE 中看起来像这样：*../../../include*。

手动导入生成代码

- 添加一个指向 cydsn 文件夹(PSoC™ Creator 放置用户文件)的路径。

如果从空项目开始，请添加 cydsn 文件夹中的文件，并设置此文件夹的路径。

4.3.2 从何处获取 PDL 库文件

库文件是 PDL 文件的副本，用户永远不应更改。从 *cydsn / Generated_Source / PSoC6 / pdl* 文件夹添加这些文件。PSoC™ Creator 仅复制所需的库文件。

PDL 模板项目具有指向 PDL 头文件的预设路径，这些文件指向 PDL 安装中的原始文件，而不是生成的代码。如果您从 PDL 模板项目开始，您应该：

- 删除 PDL 安装的任何项目相对路径。
- 替换为 *cydsn / Generated_Source / PSoC6 / pdl* 文件夹中相同位置的路径。

预设路径可以包括*/ cmsis / include*，*/ ip* 和*/ drivers / peripheral* 等。

4.4 支持迭代开发

导入生成的代码并设置项目选项以便在 IDE 中构建项目后，您将需要更改设计。将文件导入 IDE 需要耗费一定工作量，但这是一次性任务。

支持迭代开发的关键要求很简单：从 cydsn 文件夹导入文件。在 PSoC™ Creator 项目中进行设计更改后，构建代码。根据更改，代码生成过程可以修改，添加或删除文件。

修改过的文件

PSoC™ Creator 仅修改设计文件。如果将生成的代码从 cydsn 文件夹添加到项目文件中，则下一个版本将使用最新版本的文件。在 IDE 中重建代码以更新可执行文件。无需额外的工作。

添加或删除文件

将任何新文件 (设计文件或库文件) 添加到项目中。同样，如果不再需要文件，则可以将其从项目文件中删除。然后在 IDE 中重建代码以更新可执行文件。

4.5 手动导入审核

虽然本节提供了有关 PSoC™ Creator 生成代码的重要详细信息，但将生成的代码手动导入 IDE 项目文件的过程非常简单：

1. 确定所需的生成文件 (用户，设计和库)。
2. 将它们添加到项目文件中 (从它们在 cydsn 文件夹中的位置)。
3. 设置包含路径以查找头文件。

5 手动导入生成的代码 - 示例

本节将引导您完成导入从相对复杂的代码示例生成的代码的过程。由于每个项目和 IDE 都不同，因此此示例不是一系列精确的步骤和指导。

要从此演练中获得最大价值，您应下载、安装和构建 (使用 PSoC™ Creator) **CE21736 代码示例**。在 IDE 中创建一个新的空项目。然后在导入代码时浏览 PSoC™ Creator cydsn 文件夹的内容。

一个例子不能涵盖所有可能性。变量包括项目文件的选择 (模板或空)，要使用的 IDE (许多可能性)，固件本身的性质以及构建环境的详细信息。

手动导入生成代码中的信息应使您能够根据自己的情况做出明智的决策。鉴于有多种选择，每个选择都会产生不同的示例，本演练基于以下选项：

- 空项目文件 - 此方法支持任何 IDE。
- KeilµVision 工具 - 虽然这是一个受支持的 IDE，但这种选择提供了一种对比，允许用户确定哪种路径更适合他们的环境，CMSIS Pack 或手动导入。
- CE212736 作为示例项目 - 因为它提供了一个丰富的域，用于探索导入代码时所面临的选择。
- CE212736 - 具有蓝牙低功耗连接功能的 PSoC™ 6 MCU 使用 Bluetooth® LE 堆栈，汇编语言源代码和预编译二进制文件以及多个外设驱动程序。本演练不会在硬件上运行代码示例或解释其功能。

本演练将代码导入 CM4 项目。对于多 CPU 设备，您可以构建 CM0+和 CM4 项目。

完成设计并构建 PSoC™ Creator 项目后，每个 CPU 的过程都是相同的：

1. 设置 IDE 项目文件。
2. 将文件添加到项目中。
3. 设置包含路径。
4. 在 IDE 中构建项目。

5.1 完成设计并生成应用文件

在导入 IDE 之前，您必须能够在 PSoC™ Creator 中成功生成应用程序。**Figure 7** 显示了生成 CE212736 项目应用文件后的 PSoC™ Creator 工作区资源管理器。

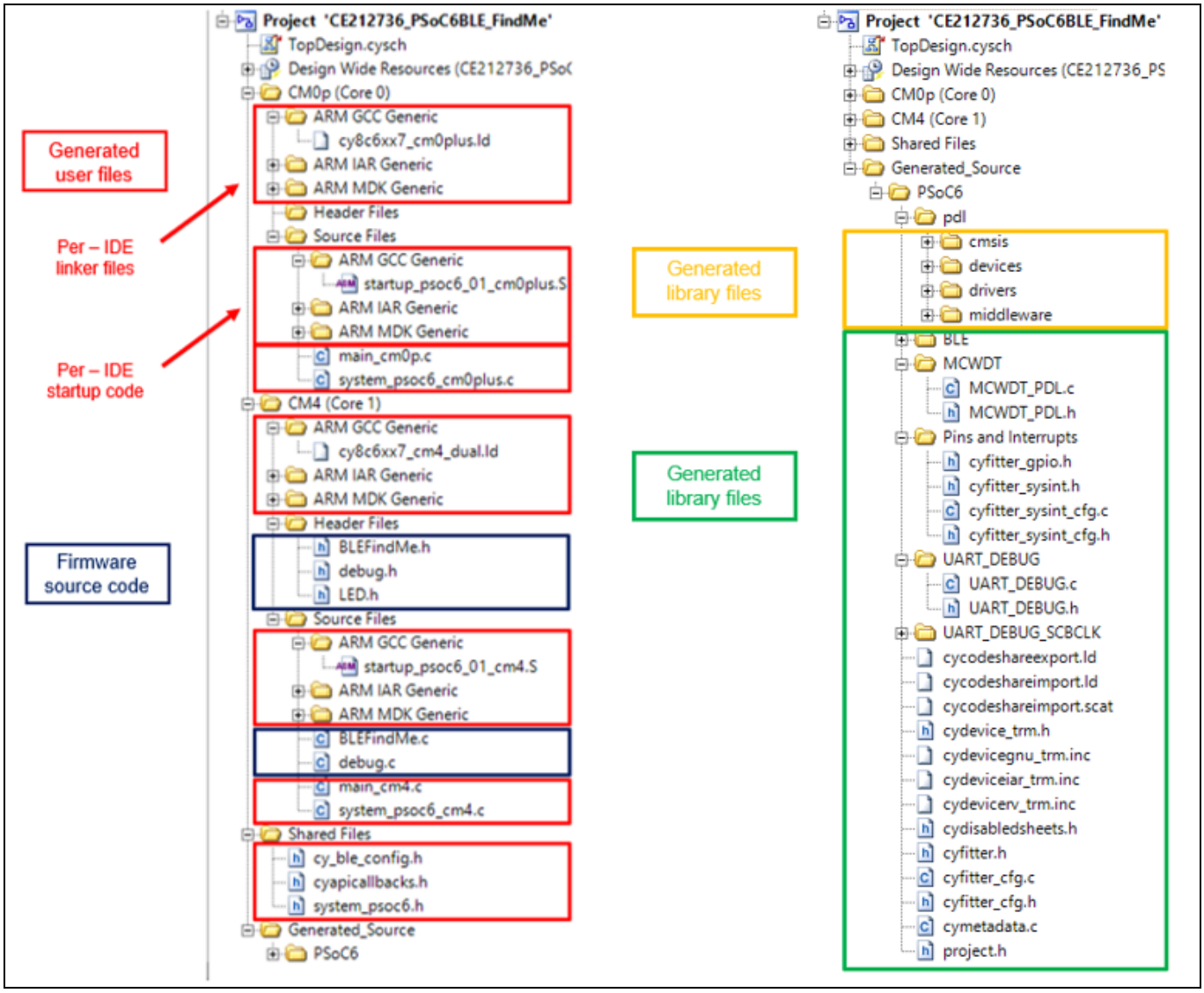


Figure 7 PSoC™ Creator 工作区资源管理器

Workspace Explorer 窗格中的 Generated_Source 项包含项目的 *cydsn / Generated_Source* 文件夹中的所有文件。该设计使用一个 Bluetooth® LE 组件、一个多计数器看门狗定时器 (MCWDT) 和一个 UART 将调试消息打印到终端窗口。它还具有控制 LED 的引脚，并设置系统中断。

因为这是一个完整的代码示例，所以该项目还包含实现该示例的固件源代码。如果您更喜欢在另一个 IDE 中开发代码，通常会在 IDE 中创建这些文件，而不是在 PSoC™ Creator 中创建。

5.2 设置 IDE 项目文件

创建一个新项目。通常，IDE 要求您指定目标设备。IDE 的项目创建过程中可能还有其他步骤。

例如，在 KeilµVision 工具中，选择 **Project** > **New µVision Project**。指定目标设备。您可以从 **Cypress GitHub repository** 下载并安装 PSoC6_DFP 包。该包在 Keil µVision 和其他支持 CMSIS 包的 IDE 中提供 PSoC™ 6 器件支持。使用正确的器件会自动设置内存地址范围、编程算法等。如果设备不可用，或者您使用的 IDE 不支持 CMSIS 包，您可以先为项目选择合适的 Arm® CPU，然后手动提供设备所需的特定细节。

指定器件后，选择软件包。**Figure 8** 显示了一个用例。在这个示例中，不需要软件包。

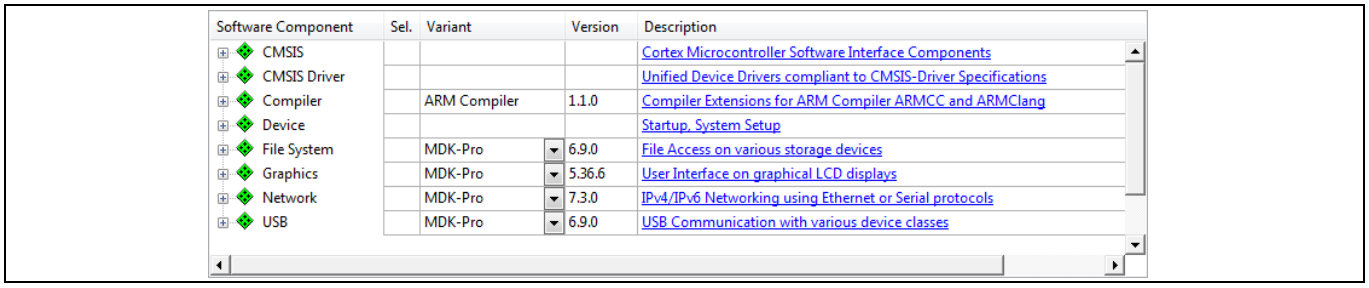


Figure 8 µVision 软件包选择

完成后，将出现一个空项目，如 **Figure 9** 所示。对于此示例，我们将项目命名为“CM4 Project”。导入多 CPU 设备的代码时，每个 CPU 都需要一个项目。

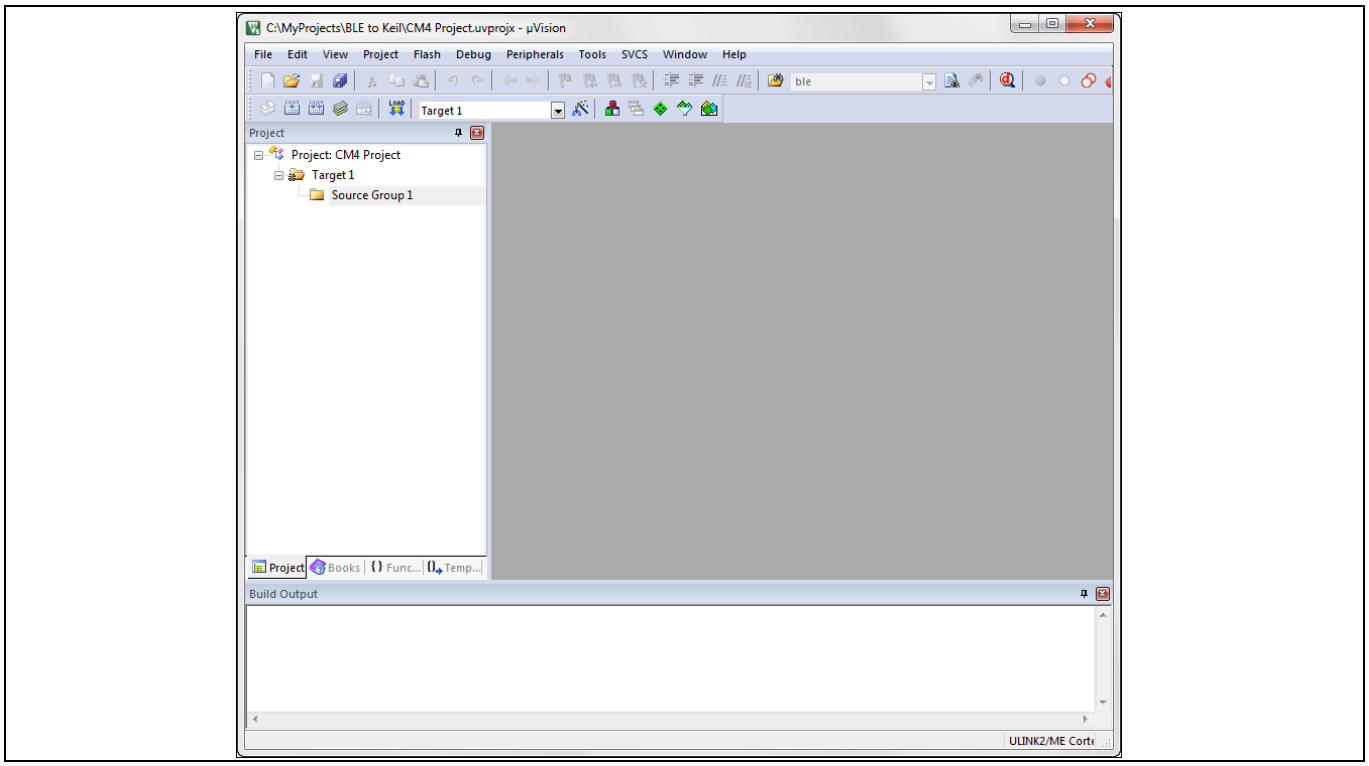


Figure 9 µVision IDE 中的空项目

5.3 添加文件到项目中

确定所需的文件，并将它们添加到项目中。每个 IDE 都有自己的添加文件工作流程。

IDE 通常具有某种文件浏览器窗格。您可以在项目中添加或删除组，以根据需要组织文件。本节中的指导仅仅是：指导。您可能更喜欢其他组织方案。**Figure 10** 显示了本演练中使用的组结构。它与 PSoC™ Creator 生成的文件类型相匹配，但有一个例外。因为 Bluetooth® LE 堆栈是一个大型文件集合，所以此示例将它们放在一个专用组中。

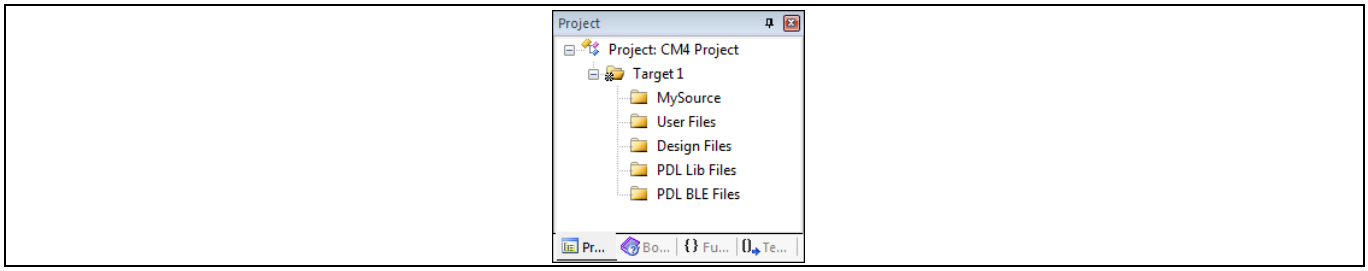


Figure 10 µVision IDE 中的任意分组结构

5.3.1 添加固件文件

因为本演练导入了一个功能齐全的代码示例，我们将固件文件导入 MySource 组，如 **Figure 11** 所示。固件文件都在 PSoC™ Creator 项目的 cydsn 文件夹中。在您自己的工作中，您将创建自己的固件文件，并在合适的项目文件中组织它们。

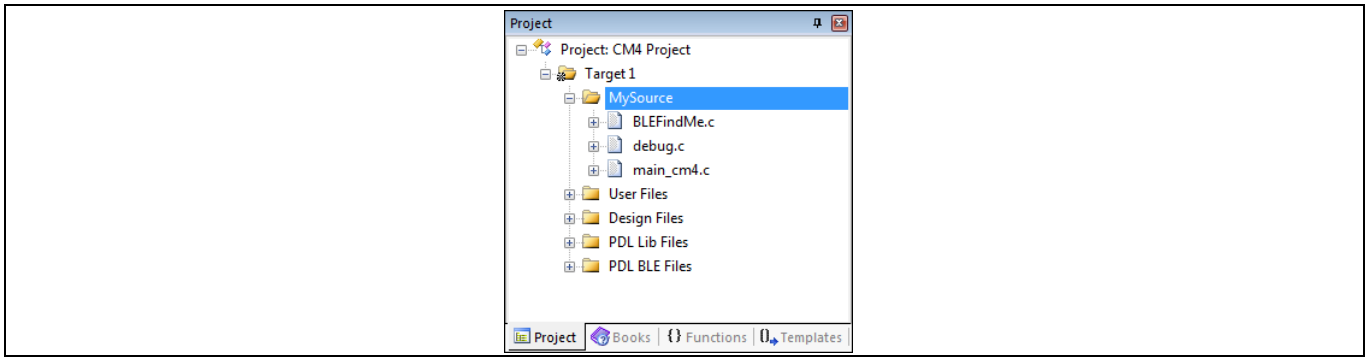


Figure 11 添加到项目中的固件文件

Note: 为 CM0+项目重复此过程时，请添加特定于该 CPU 的文件。

5.3.2 添加用户文件

用户文件由 PSoC™ Creator 生成，但您可以更改这些文件。用户文件位于 cydsn 文件夹的顶层。启动代码位于特定于 IDE 的文件夹中。请参阅**用户文件**。在这种情况下，用户文件是：

- 系统配置文件 (特定 CPU)
- 启动文件 (特定 IDE 和 CPU)

Figure 12 显示了项目资源管理器中的用户文件。

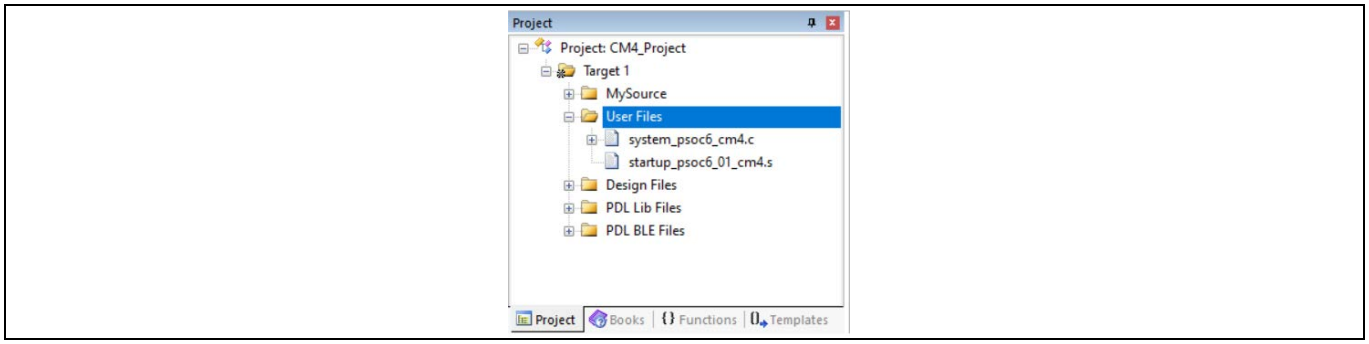


Figure 12　添加到项目的用户文件

Note:　启动文件是特定于 IDE 的汇编程序源代码。生成的代码包含所有支持的 IDE 的启动代码。使用您需要的文件。如果您使用的是不受支持的 IDE，请提供您自己的启动代码。

5.3.3　添加设计文件

设计文件是 PSoC™ Creator 生成的源文件和头文件，专门针对您的系统设计及其组件。请参见**设计文件**。设计文件位于 *cydsn / Generated_Source / PSoC6* 文件夹中。

Figure 13 显示了添加到 µVision 项目的此项目的设计文件。其中包括 cyfitter 和 cymetadata 文件，以及 Bluetooth® LE，MCWDT 和 UART 组件的特定于组件的源文件。

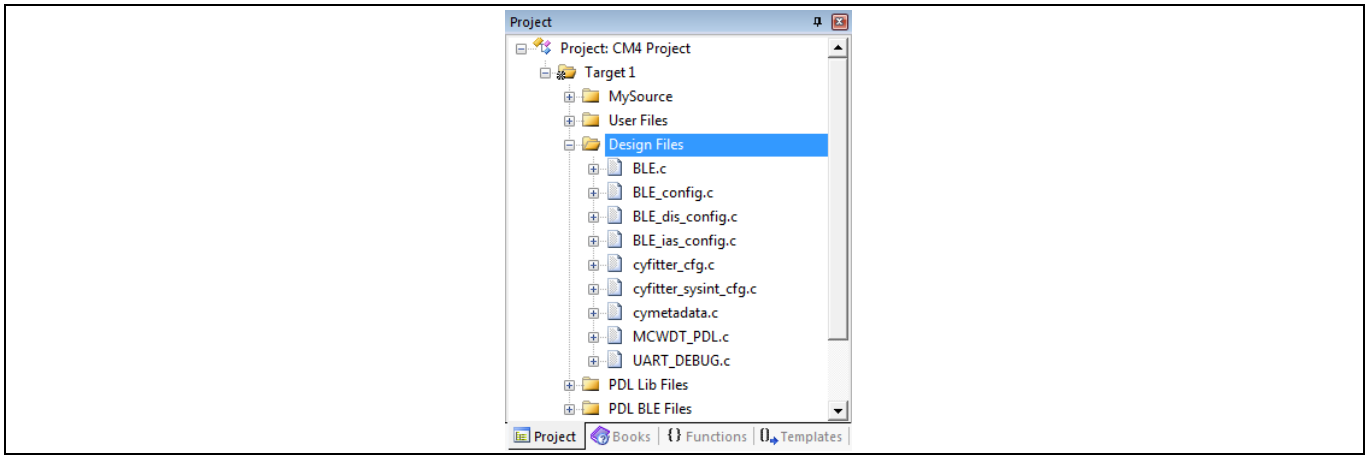


Figure 13　添加到项目的设计文件

5.3.4 添加 PDL 库文件

库文件从 PDL 安装中复制。请参阅**库文件**。根据您的设计，PSoC™ Creator 将所需文件复制到 *cydsn / Generated_Source / PSoC6 / pdl / drivers / peripheral* 文件夹中。**Figure 14** 显示了添加到 IDE 的此项目的外设驱动程序文件。

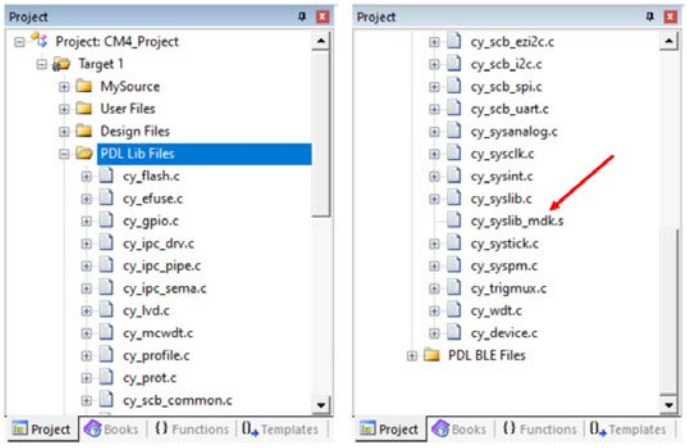


Figure 14 添加到此项目的库文件

Note: *系统库 (syslib) 的实现包括特定于 IDE 的汇编程序文件，如 **Figure 14** 所示。如果未将该文件添加到项目中，则构建将失败。*

Note: *每个驱动程序在 peripheral 文件夹中都有自己的子文件夹。在文件夹结构中上下导航时，添加源文件可能很繁琐。如果 IDE 支持通过拖放添加文件，则可以使用 Windows 资源管理器使任务更容易。转到 peripheral 文件夹，然后搜索***.c**。所有.c 文件如 **Figure 15** 所示。然后可以将它们拖到 IDE 中。(不幸的是，这个技巧不适用于 µVision IDE。) 不要忘记添加汇编源文件。*

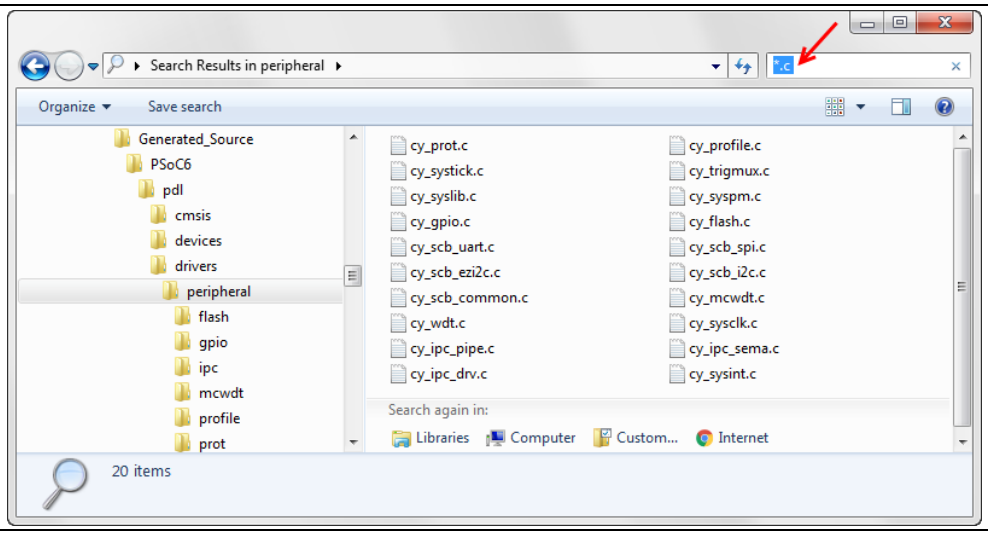


Figure 15 查找.c 文件

5.3.5 添加 Bluetooth® LE 库文件

此设计中使用的最终文件集是 Bluetooth® LE 堆栈。生成的代码位于 *cydsn / Generated_Source / PSoC6 / pdl / middleware / ble* 文件夹中。Bluetooth® LE 堆栈的某些部分是作为特定 CPU 的二进制库提供的，而不是源代码。必须将所有必需的源文件和二进制库添加到项目中。

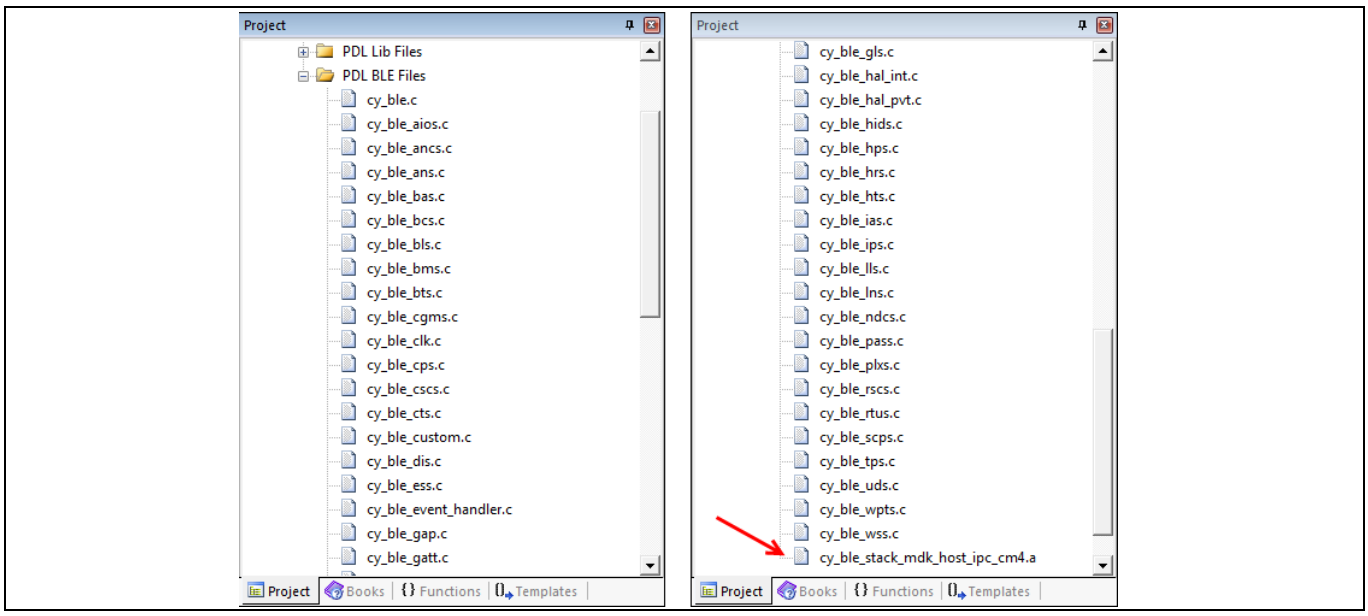


Figure 16 添加到项目中的 Bluetooth® LE 堆栈文件

Note: 为 CM0+项目重复此过程时，请添加特定于该 CPU 的库。

*Note: IDE 可能会将库文件视为汇编源文件而不是库，这会导致生成错误。IDE 可能具有控制此问题的属性。例如，**Figure 17** 显示了 µVision IDE 的文件属性对话框。*

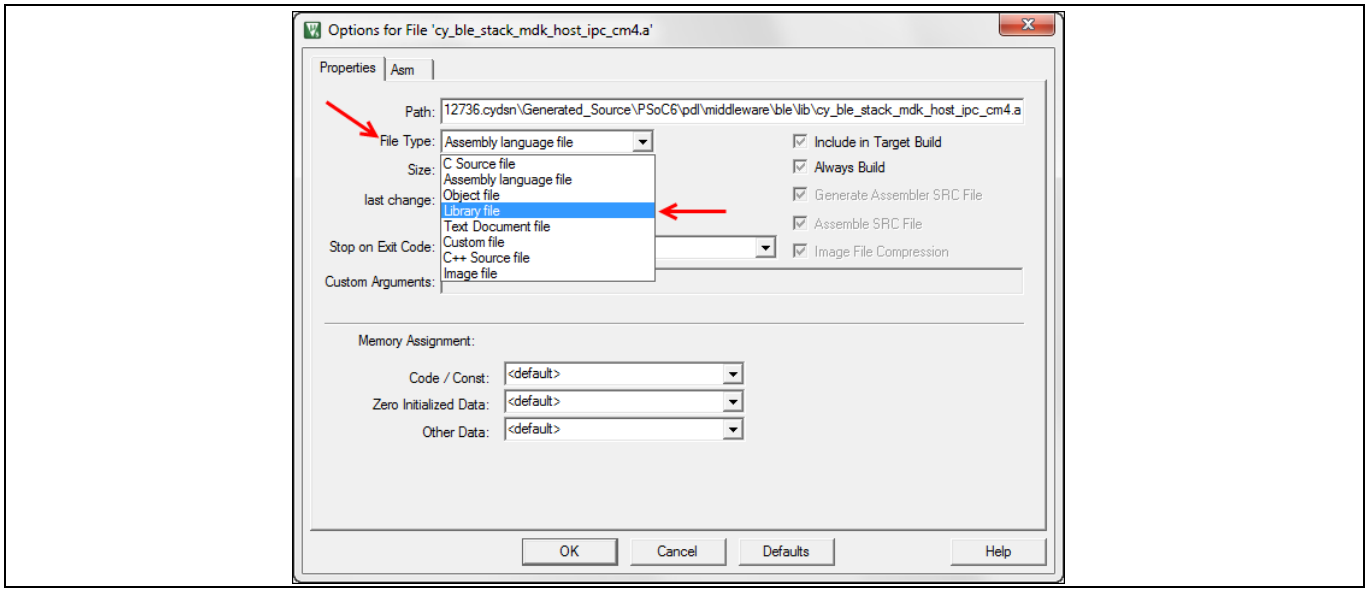


Figure 17 在 µVision IDE 中设置文件类型

5.4 选择包含路径

IDE 的构建系统必须找到所有必需的头文件。一些开发人员更喜欢将头文件直接添加到项目浏览器中以便于访问。但是，这通常不会为 IDE 设置包含路径。

每个 IDE 都有自己的设置包含路径的方法。此示例假定您知道如何在首选 IDE 中添加包含路径。本节的目的是向您展示需要添加的路径，而不是如何添加它们。

在 µVision IDE 中，您可以在 **Include Paths** 项目的目标选项 C / C++面板中添加路径。

5.4.1 设置用户文件路径

PSoC™ Creator 生成的用户文件的所有头文件都位于 cydsn 文件夹中。当然，路径取决于您放置 PSoC™ Creator 项目的位置。在此示例中，CE212736 项目的路径可能如下所示：

C:/MyProjects/CE212736/CE212736.cydsn

Figure 18 显示了在 µVision IDE 中设置该路径。

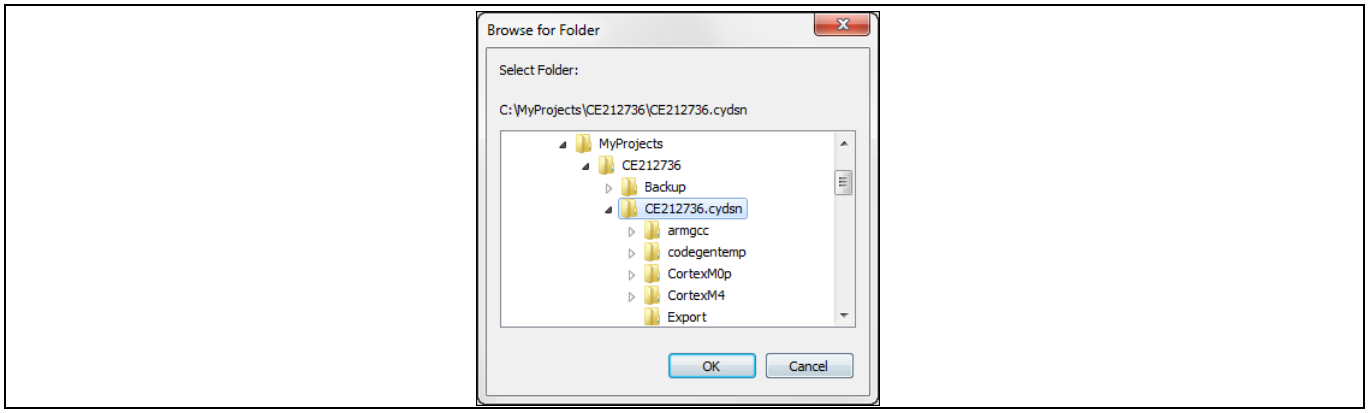


Figure 18 设置 *cydsn* 文件夹的路径

Note: 在此示例中，固件头文件也位于此文件夹中，因此不需要为它们提供单独的路径。

5.4.2 设置设计文件的路径

PSoC™ Creator 生成的设计文件的所有头文件都位于 *cydsn / Generated_Source / PSoC6* 文件夹中。在此示例中，CE212736 项目的路径可能如下所示：

C:/MyProjects/CE212736/CE212736.cydsn/Generated_Source/PSoC6

Figure 19 显示了在 µVision IDE 中设置该路径。

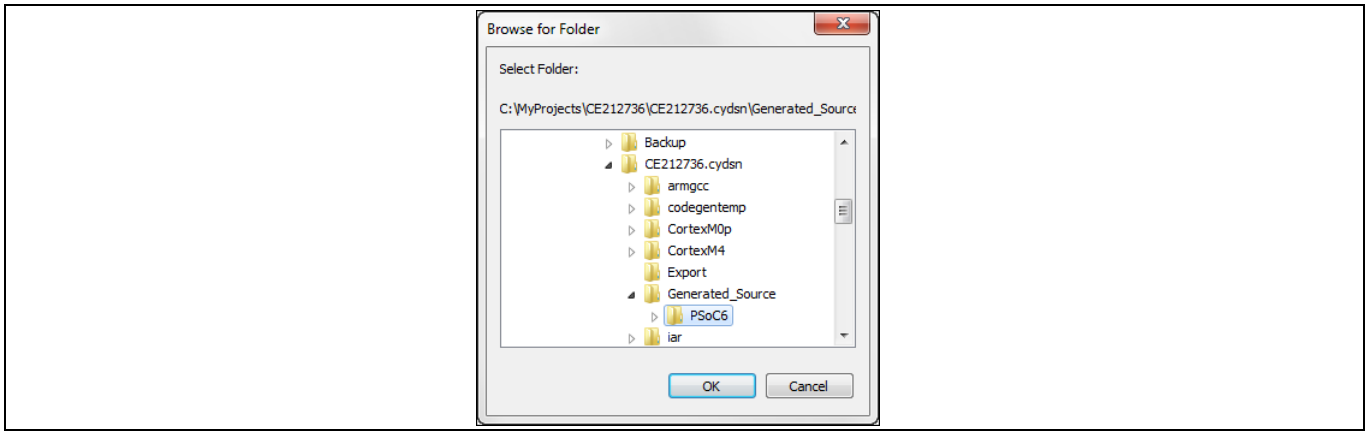


Figure 19 设置设计文件的路径

5.4.3 设置外设文件夹的路径 (驱动程序文件)

每个外设驱动都有自己的子文件夹，并且每个外设驱动都有一个头文件，需要包含在项目中。所以，需要将每个驱动子文件夹的路径作为 Include Paths 的一部分添加进去。例如，在这个项目中包含 flash 驱动头文件所要添加的路径是：

C:/MyProjects/CE212736/CE212736.cydsn/Generated_Source/PSoC6/pdl/drivers/peripheral/flash

此外，还应该添加包含所有驱动子文件夹的外设文件夹的路径。CE212736 项目的外设文件夹路径为：

C:/MyProjects/CE212736/CE212736.cydsn/Generated_Source/PSoC6/pdl/drivers/peripheral

Figure 20 显示了 µVision IDE 中外设文件夹路径的设置。

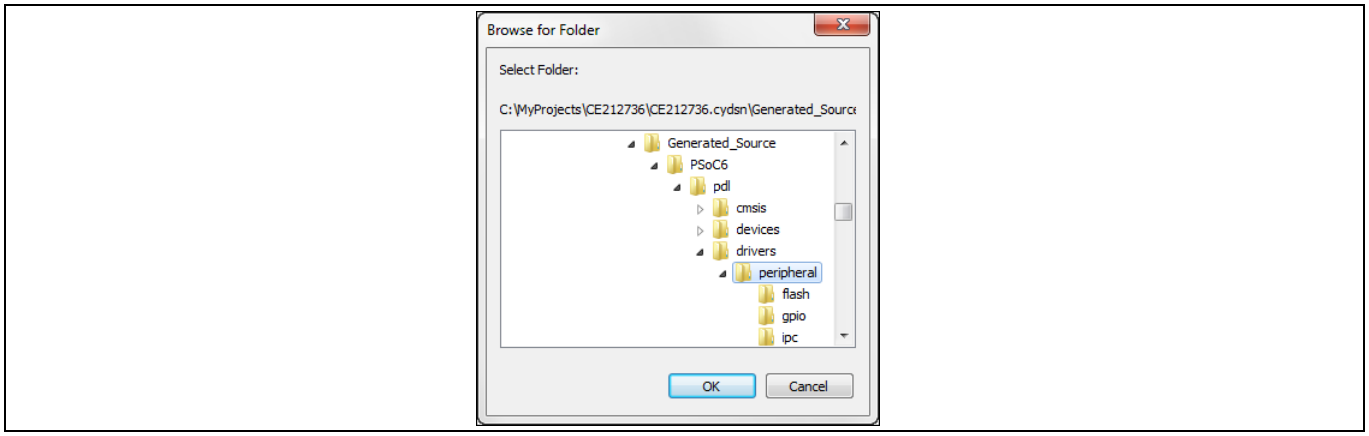


Figure 20 设置外设文件夹的路径

5.4.4 设置中间件文件夹路径(Bluetooth® LE 堆栈)

虽然 Bluetooth® LE 堆栈位于子文件夹中，但源代码提供了在中间件文件夹中包含任何头文件的路径。因此，单个路径适用于此文件夹树中的所有位置。在此示例中，CE212736 项目的路径可能如下所示：

C:/MyProjects/CE212736/CE212736.cydsn/Generated_Source/PSoC6/pdl/middleware

Figure 21 显示了在 µVision IDE 中设置该路径。

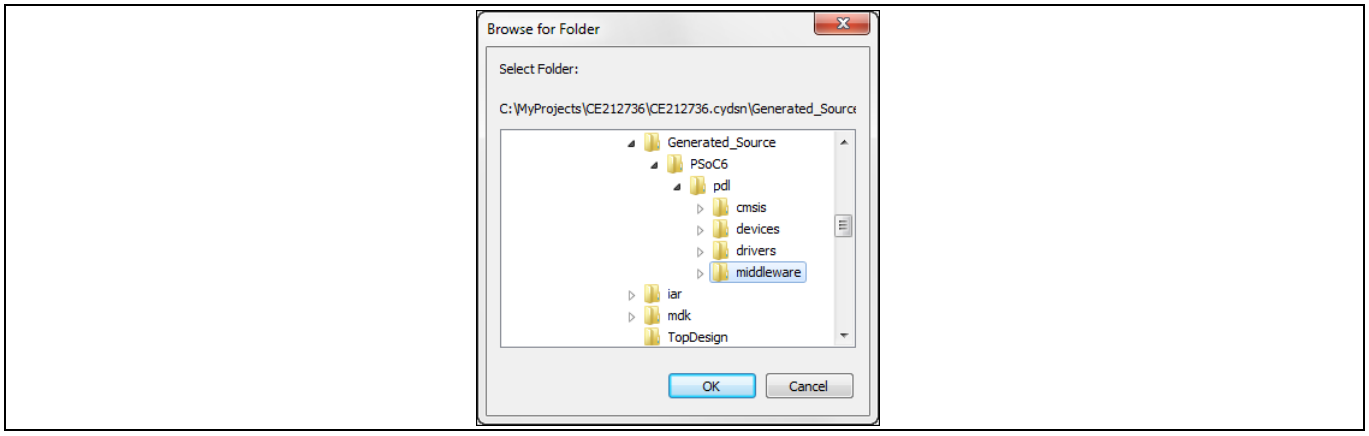


Figure 21 设置 Bluetooth® LE 堆栈路径

5.4.5 设置其它需要的路径

成功构建必须包含其他头文件。PSoC™ Creator 在 *cydsn / Generated_Source / PSoC6 / pdl* 文件夹中提供这些头文件的副本：

- CMSIS 头文件: *cydsn/Generated_Source/PSoC6/pdl/cmsis/include*
- IP 头文件: *cydsn/Generated_Source/PSoC6/pdl/devices/psoc6/include/ip*
- 特定系列头文件: *cydsn/Generated_Source/PSoC6/pdl/devices/psoc6/include*

Figure 22 显示了 µVisionIDE 中设置的这三个路径，指向生成的代码。

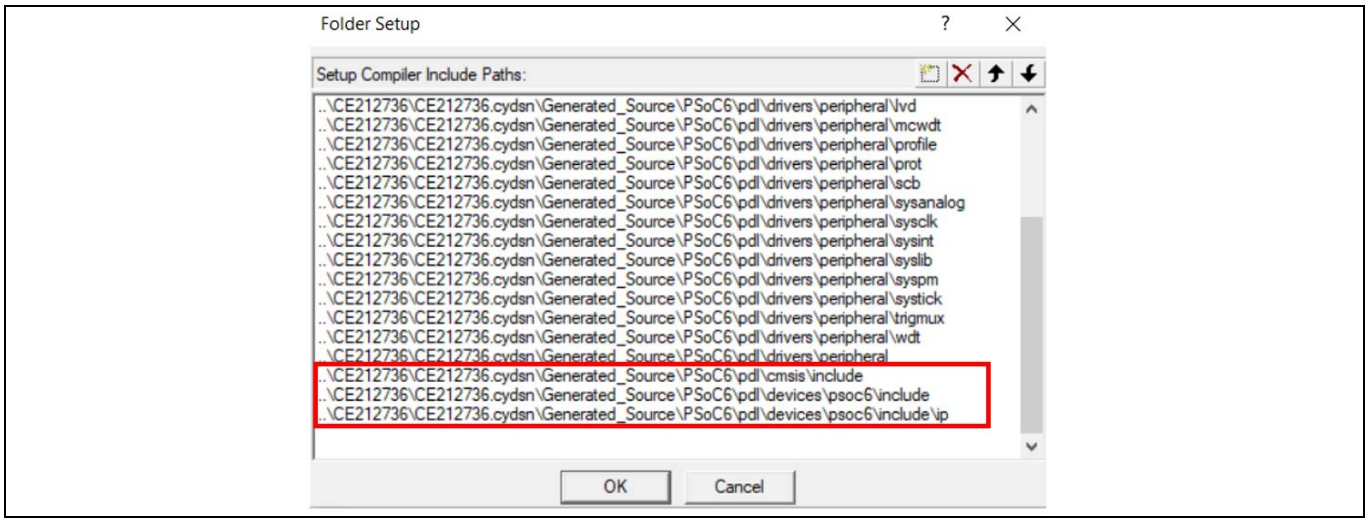


Figure 22 其它需要的头文件的路径

Note: *如果从 PDL 模板项目开始，可能已经为您设置了这三个路径。但是，它们指向 PDL 安装中的原始文件，而不是生成的代码。您应该修改这些路径以指向 cydsn 文件夹中的相应位置。请参阅**从何处获取 PDL 库文件**。*

5.5 在 IDE 中构建项目

在编译代码之前，必须配置应用程序成功构建所需的所有选项。本应用笔记介绍如何将生成的代码导入 IDE，而不是如何在任何给定的 IDE 中配置项目。但是，**设置 IDE 项目文件**可提供其他背景信息。

添加所有必需文件并正确设置所有路径后，构建项目。它应该成功构建。您可能会遇到警告或错误。每个 IDE 都有一个唯一的界面、以及唯一的选项、默认设置以及警告和错误消息。由于 IDE 的可变性，本应用笔记无法提供有关处理错误的详细指导。您熟悉 IDE 将大大有助于解决您遇到的任何问题。

但是，与导入代码相关，通常会遇到三种错误。

如果头文件发生“找不到文件”错误，请在文件系统中找到实际文件。确保该文件有一个包含路径。

如果未定义函数或符号，请确保已将所有用户、设计和库文件添加到项目中，并且找到声明该符号的任何头文件。例如，未能包含二进制库会导致此类错误。

即使所有文件和路径都正确，您也可能会遇到一连串的编译器或链接器错误。您的项目中可能没有正确配置设置。例如，如果 IDE 中的编译器不支持默认支持 C99 标准，请确保启用了支持。

5.6 示例评审

本节提供了将代码手动导入 IDE 的示例。虽然单个示例无法涵盖所有情况，但任务很简单：

1. 在 IDE 中创建和配置项目文件，对于多 CPU 设备，每个 CPU 一个。或者，从 PDL 模板项目开始。
2. 将所需的 PSoC™ Creator 生成的文件添加到项目文件中 (对于多 CPU 设备，为每个 CPU 执行此操作)。
3. 设置这些文件的包含路径。

PDL 模板项目提供闪存配置、链接器和启动文件。如果您更喜欢不受支持的 IDE，请将它们用作参考。特定于 IDE 的文件位于此处：*<PDL Install Folder>/devices/psoc6*.。

最后，本应用笔记向您介绍了导入代码的基本原则，例如存在哪些文件，在何处查找它们以及如何使用它们。您需要将这些知识应用于您的情况。

6 总结

为 PSoC™ 6 MCU 等双 CPU 嵌入式系统编写软件可能是一项艰巨的任务。PSoC™ Creator 简化了这个过程。您可以使用友好的 UI 创建和配置设计。PSoC™ Creator 生成实现该设计所需的所有代码，只需点击一下按钮即可。您专注于真正的价值，在生成的代码之上创建固件。

您可以完全在 PSoC™ Creator 中开发该固件，但许多开发人员和组织都有一个更习惯的开发系统。

本应用笔记介绍了如何在第三方 IDE 中使用 PSoC™ Creator 生成的代码：通过导出和导入支持的 IDE，或通过手动导入所需文件的任何 IDE。您了解了不同类型的生成文件、查找文件的位置以及如何将它们添加到 IDE 的项目中。

本应用笔记中包含的知识使您能够结合两者的优点：高质量生成的代码以缩短开发时间，以及您首选的 IDE。

7 相关文档

应用笔记	
AN210781 – 具有蓝牙低功耗 (Bluetooth® LE) 连接功能的 PSoC™ 6 MCU 入门	介绍带有 Bluetooth® LE 连接器件的 PSoC™ 6 MCU 以及如何构建您的第一个 PSoC™ Creator 项目
AN215656 – PSoC™ 6 MCU：双核 CPU 系统设计	介绍 PSoC™ 6 MCU 中的双核 CPU 架构，并展示如何构建简单的双 CPU 设计
AN225588 –ModusToolbox 软件与第三方 IDE 结合使用	描述了 ModusToolbox 软件和配置器的类似过程，而不是 PDL 和 PSoC™ Creator 的过程
PSoC™ 6 MCU	
PSoC™ 6 MCU 主页	提供对所有 PSoC™ 6 MCU 资源的访问
PSoC™ 6 MCU 社区	讨论 PSoC™ 6 MCU 的问题
外设驱动程序库 (随 PDL 一起安装的文档)	
外设驱动库 v3.0 用户指南	库概述以及如何使用
外设驱动库 API 参考	API 的详细技术参考
PSoC™ Creator	
PSoC™ Creator 产品页	访问下载，培训，组件等
PSoC™ Creator 快速指南	快速启动并运行
PSoC™ Creator 用户指南	综合手册
开发套件文档	
CY8CKIT-062-BLE PSoC™ 6 Bluetooth® LE Pioneer Kit	

8 附录 A. 配置 IDE 项目文件

要使代码成功构建，必须使用各种选项的设置配置 IDE 项目文件。由于每个 IDE 都具有唯一的 UI 和默认设置，因此如何为任何特定 IDE 配置空项目超出了本应用笔记的范围。但是，具有您需要配置什么的一般概念是非常宝贵的。有了这些知识，您可以在 IDE 中找到并设置适当的选项。

Table 6 列出了配置 IDE 项目文件所需的几个选项。对于任何给定的 IDE，可能还有其他选项。任何选项的默认设置可能已正确设置。

Table 6 IDE 选项

类型	选项	备注
器件	目标	如果 IDE 支持特定设备，请选择该设备。这通常会自动设置其他与设备相关的选项。如果设备未在 IDE 中列出，请选择通用 Cortex® M4 或 M0+设备，并确保正确设置所有与设备相关的选项。
编译器	C99 支持	启用对 C99 标准的支持。
	浮点	为 CM4 CPU 启用浮点支持。
	优化	设置适合您的构建的编译器优化级别。
	调试	为调试版本生成调试信息。
	包含路径	设置包含路径。本应用笔记讨论了与生成的代码相关的所有路径。
	调试标志	有条件编译的 PDL 代码需要定义调试符号。对于调试版本，请定义 DEBUG。对于非调试版本，请定义 NDEBUG。
	器件标志	有条件编译的特定于设备的 PDL 代码需要定义正确的符号。此符号控制用于构建的特定于设备的头文件。请参阅 *cy_device_headers.h*。
连接器	命令文件	指定链接描述文件的路径。PDL 为受支持的 IDE 提供链接器文件。如果 IDE 具有对特定设备的支持，则可以自动设置。
	其它设置	为您的版本设置其他链接器选项。例如，生成链接器映射文件，输出调试信息，生成日志文件等。
调试器	调试连接	指定支持的调试连接 (探测)，例如 J-Link 或 CMSIS-DAP。
	连接设置	根据您的调试连接，指定各种连接设置，例如重置选项，连接速度，缓存选项，下载选项等。
	内存配置	指定设备的内存区域。如果 IDE 具有特定于设备的支持，则可以自动设置。
	寄存器描述	PDL 为寄存器级调试信息提供系统视图描述 (SVD) 文件。如果 IDE 具有对特定设备的支持，则可以自动设置。
	闪存加载器	指定用于将可执行文件下载到设备的 flashloader。如果 IDE 具有对特定设备的支持，则可以自动设置。

9 附录 B. 将生成代码用于学习

即使您因环境无法导入 PSoC™ Creator 生成的代码，该代码仍然是协助固件开发的宝贵资源。

在这种情况下，PSoC™ Creator 生成的代码的主要价值是作为如何使用 PDL 的学习资源。有几个方面可以提供重要帮助，包括但不限于：

- 时钟配置– 参阅 *system_psoc6_cm0plus.c*, *system_psoc6_cm4.c*, and *cyfitter_cfg.c*
- 中断配置 – 参阅 cyfitter_sysint_cfg.c
- 引脚配置 – 参阅 cyfitter_gpio.h and cyfitter_cfg.c
- 外设配置 – 参阅特定组件的*.c* 和*.h* 文件
- PDL 功能 API – 参阅特定组件的*.c* 和*.h* 文件

在每种情况下，您都可以从生成的代码中提取代码片段、特定函数、算法甚至完整文件，并在您自己的代码中使用它们。例如，您可以选择复制外设的配置结构，或在编写自己的代码时参考这些结构。

PSoC™ Creator 在 PDL API 之上生成特定于组件的 API。组件 API 通常具有定义的函数，该函数将 1：1 映射到 PDL API。组件 API 根据设计提供所需的硬件参数。它通常还包括 `Start()`和 `Stop()`函数，这些函数用于初始化、启用或终止特定外设所需的 PDL API 调用 (按正确顺序)。您可以浏览组件 API 以了解它如何使用 PDL API。

此外，由于大多数 PDL 都是作为源代码提供的，因此您可以浏览 PDL 源文件以查看用于控制功能和行为的寄存器。

各种生成的代码文件之间存在依赖关系。代码生成过程定义符号并使用其自己的命名约定。它为每个组件创建完整的 API。您可以决定直接使用多少，以及多少用于适应固件开发过程。

文档修订记录

版本	提交日期	变更说明
**	2019-01-10	翻译自 002-19434 Rev. *B
*A	2020-01-20	翻译自 002-19434 Rev. *C
*B	2021-01-22	翻译自 002-19434 Rev. *D
*C	2022-06-27	翻译自 002-19434 Rev. *E;更新至 Infineon 模板

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

Edition 2022-06-27
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
002-26111 Rev. *C

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。