AN73854

PSoC® 3、PSoC 4 和 PSoC 5LP 中 Bootloader 的简介

作者： Mark Ainsworth

相关器件系列：所有 PSoC 3、PSoC 4 和 PSoC 5LP 器件

相关应用笔记:要想获取完整的列表，请点击此处。

如果需要本应用笔记的最新版本，请访问 http://www.cypress.com/AN73854。

AN73854 简单介绍了 Bootloader 的理论和技术。然后说明了如何使用 PSoC Creator™在 PSoC® 3、PSoC 4 和 PSoC 5LP MCU 中快速轻松地实现 Bootloader。各主题包括 Bootloader 系统的说明、性能和自定义程序。

目录

1 简介 2
2 PSoC 资源 3
3 PSoC Creator 3
4 Bootloader 是什么？ 4
4.1 术语和定义 4
4.2 使用 Bootloader 5
4.3 Bootloader 功能流程 5
5 通用 Bootloader 设计注意事项 6
5.1 Bootloader 的替代方法 6
5.2 存储器使用情况和模块配置 6
5.3 Bootloader — 主机时序 7
5.4 通信端口 7
5.5 从失败中恢复 7
5.6 将来可靠性保证 8
5.7 自定义程序 8
6 PSoC Bootloader — 工作原理 9
6.1 PSoC Creator Bootloader 项目 9
6.2 Bootloader 选项 10
6.3 通信组件 10
6.4 从故障恢复 10
6.5 向后兼容 10
6.6 Bootloader 存储器的使用情况 10
6.7 Flash 保护 10
6.8 自定义程序 11
7 将 Bootloader 添加到 PSoC Creator 项目中 12
7.1 建立一个 Bootloader 12
7.2 添加 Bootloadable 应用 16
7.3 调试 Bootloadable 项目 16
7.4 自定义 Bootloader 18
7.5 调用 Bootloader 19
8 将您的项目加载到 PSoC 内 19
8.1 项目文件 19
8.2 使用情况 20
9 双应用 Bootloader 的注意事项 21
9.1 应用启动流程 23
10 总结 24
11 相关应用笔记 24
A 附录 A — Bootloader 和器件复位 25
A.1 为什么需要执行器件复位？ 25
A.2 对器件 I/O 引脚的影响 25
A.3 对其他功能的影响 27
A.4 示例：风扇控制 27
文档修订记录 28
全球销售和设计支持 29

1 简介

本应用笔记简单介绍了 Bootloader 的基本原理和设计原理，然后说明了如何在 PSoC 3、PSoC 4 和 PSoC 5LP 的 PSoC Creator 项目中实现 Bootloader。

如果您刚开始接触 Bootloader，则可以在 Bootloader 是什么？和通用 Bootloader 设计注意事项章节中找到 Bootloader 的基本概念和设计原理。

如果您已经熟悉了 Bootloader，并且想了解如何在使用 PSoC Creator 的 PSoC 3、PSoC 4 和 PSoC 5LP 器件中实现它，请参考 PSoC Bootloader — 工作原理章节中的内容。

要想了解如何将 Bootloader 添加到您的 PSoC Creator 项目中，请参考将 Bootloader 添加到 PSoC Creator 项目中的内容。

要想获取与 I^2C、UART、SPI 和 USB 相关的 Bootloader 应用笔记列表，请参考相关应用笔记章节。该章节中列出的每个 Bootloader 应用笔记都有相应的代码示例。

通过菜单选项 **File**（文件）> **Example Project**（示例项目），您可以从 PSoC Creator 中访问与示例项目相关的 Bootloader。在弹出的窗口中搜索“bootloader”，从而可以滤除与项目相关的 Bootloader。

请点击此处获取 PSoC 3、PSoC 4 和 PSoC 5LP 代码示例的完整列表。

注释： 从 PSoC Creator 2.1 开始，已经重新组织了 Bootloader 系统以提供更多的配置选项。在以前发布的版本中，Bootloader 系统是 cy_boot 组件的一部分（在所有设计中会无形中自动得到实例化的必需组件）。现在，Bootloader 功能可作为单独组件使用。要想将 Bootloader 项目的旧版本移植到 PSoC Creator 2.1 SP1 或更高版本中，请参考*系统参考指南*中的第 11 章节（**Help**（帮助）> **Documentation**（文档）> **System Reference**（系统参考）。

2 PSoC 资源

在赛普拉斯网站 www.cypress.com 上提供了大量资料，有助于选择符合您设计的 PSoC 器件，并能够快速有效地将该器件集成到您的设计中。有关使用资源的完整列表，请参考 KBA86521 — 如何使用 PSoC 3、PSoC 4 和 PSoC 5LP 进行设计。下面提供了 PSoC x 的简要列表：

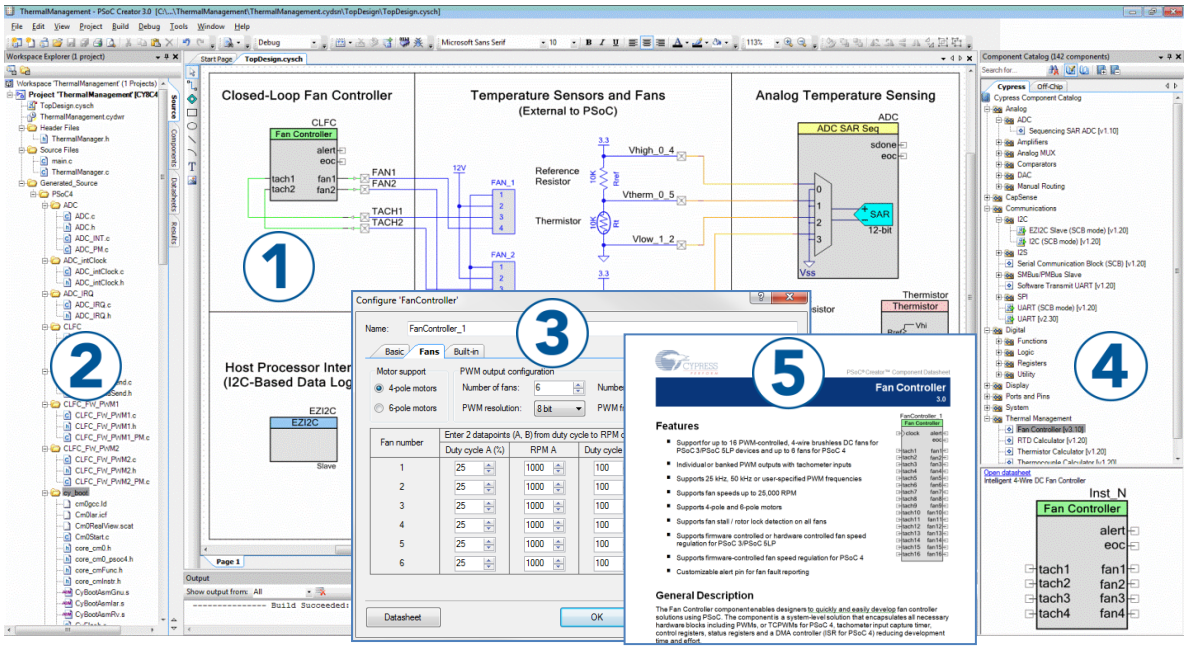
- **概况：**PSoC 产品系列、PSoC 蓝图
- **产品选型：**PSoC 1 、PSoC 3 、PSoC 4 或 PSoC 5LP。此外，PSoC Creator 还包含了一个器件选择工具。
- **数据手册：**说明并提供了适用于 PSoC 3、PSoC 4 和 PSoC 5LP 器件系列的电气规格。
- **应用笔记和代码示例：**包括从基本到高级的广泛主题。许多应用笔记还提供了代码示例。
- **技术参考手册（TRM）：**对 PSoC 3、PSoC 4 和 PSoC 5LP 器件系列中所使用的架构和寄存器进行了详细说明。
- **开发套件：**
  - CY8CKIT-001 是所有 PSoC 系列产品经常使用的开发平台。
  - CY8CKIT-030 和 CY8CKIT-050 是专门为模拟性能而设计的。通过该套件，您可以评估、开发基于 PSoC 3 和 PSoC 5LP 的高精度模拟、低功耗以及低电压应用，也可以定义这些应用的原型。
  - CY8CKIT-046、CY8CKIT-044、CY8CKIT-042 和 CY8CKIT-040 等的 PSoC 4 Pioneer 套件均为易于使用且廉价的开发平台。这些套件包括用于 Arduino™兼容屏蔽和 Digilent® Pmod™子卡的连接器。
  - MiniProg3 器件提供一个用于进行闪存编程和调试的接口。

3 PSoC Creator

PSoC Creator 是一个基于 Windows 的免费集成设计环境（IDE）。通过它可以同时在基于 PSoC 3、PSoC 4 和 PSoC 5LP 的系统中设计硬件和固件。如图 1 所示：通过 PSoC Creator，您可以进行下面的操作：

1. 将组件拖放到主要设计工作区中，以进行硬件系统设计
2. 协作设计您的应用固件和 PSoC 硬件
3. 使用配置工具配置各组件
4. 研究包含 100 多个组件的库
5. 查看组件数据手册

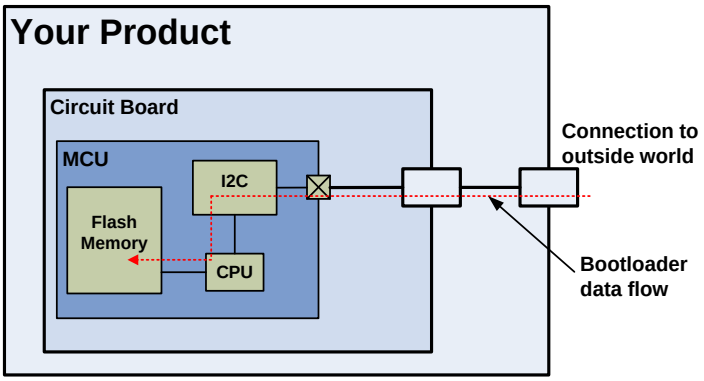
图 1. PSoC Creator 特性



4 Bootloader 是什么？

Bootloader 是 MCU 系统设计中常用的部分。通过 Bootloader，可以对产品的固件进行现场更新。在典型的产品中，固件被嵌入到 MCU 的闪存存储器内。MCU 被安装到 PCB 上，并且被嵌入到某个产品内，如图 2 所示。

图 2. Bootloader 数据流模块框图

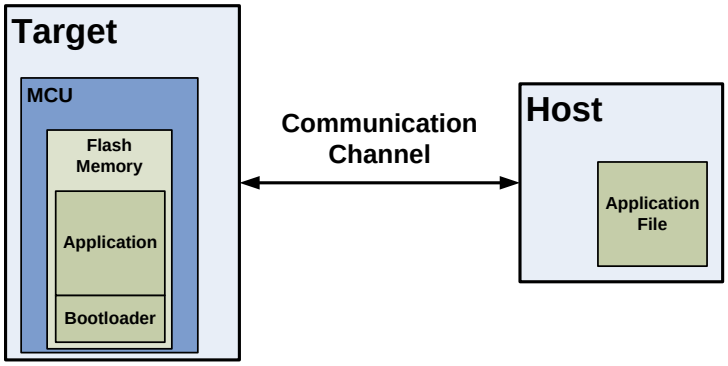


在工厂里，初始将固件编程到某个产品内时，通常是通过 MCU 的联合测试行动小组（JTAG）或串行线调试器（SWD）接口实现的。但现场经常没有这种接口 — 因此拆分产品，然后直接访问 PCB 会显得可能比较困难而且非常昂贵。一种优越的方法是使用该产品与外界间现有的连接。该连接可以是一个标准的端口，如 I^2C、USB 或 UART，也可以是一个自定义的专用协议。

4.1 术语和定义

通过图 2 可以知道，产品中的嵌入式固件完全可以通过通信端口实现两个不同的操作 — 普通操作和更新闪存操作。用于更新闪存的嵌入式固件部分被称为 **Bootloader**，如图 3 所示。

图 3．Bootloader 系统



一般情况下，用于更新闪存的数据的系统被称为**主机（Host）**，被更新的系统被称为**目标机 （Target）**。主机可以是一个外部 PC，也可以是与该目标机位于相同 PCB 上的另一个 MCU。从主机到目标闪存的数据传输过程被称为**引导加载过程**，或**引导加载操作**，或简称为**引导加载**。放置在闪存中的数据被称为**应用程序**或 **Bootloadable**。

引导加载的另一个常见名称是**系统内编程(ISP)**。赛普拉斯有一个与系统内串行编程器（ISSP）名称相似的产品，它的操作被称为主机源串行编程（HSSP）。更多有关信息，请参考第 6 页上的 HSSP。

4.2 使用 Bootloader

Bootloader 和应用程序通常共用一个通信端口。要想使用 Bootloader，首先要对产品进行操作，以执行 Bootloader，而不是应用程序。

一旦运行了 Bootloader，主机可以使用通信通道发送“start bootload”（开始引导加载）命令。如果 Bootloader 发送了一个“OK”响应，便能开始执行引导加载过程。

在引导加载过程中，主机将读取新应用程序文件，并将读取的内容解析成闪存写命令，然后把这些命令发送到 Bootloader 中。发送完整个文件后，Bootloader 将控制权交给新的应用程序。

4.3 Bootloader 功能流程

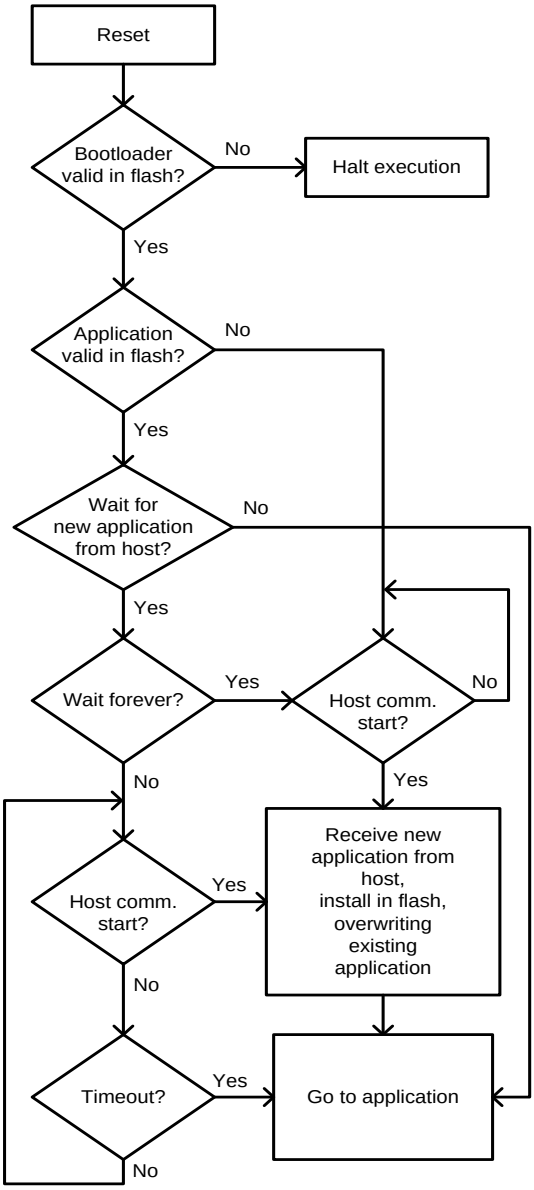
一般情况下，Bootloader 先在复位时执行（请参考第 6 页上的存储器使用情况和模块配置）。然后它将执行以下操作：

- 在运行应用程序前，先检查它的有效性
- 管理时序，以启动主机通信
- 实现引导加载 /闪存更新操作
- 最后，将控制权交给应用程序

图 4 显示的是 Bootloader 的工作流程图。

注释： PSoC Creator 支持双应用选项。在该情况下，图 4 中的“Go to application”功能以更复杂的方式运行。更多有关信息，请查阅应用启动流程。

图 4. Bootloader 功能流程



5 通用 Bootloader 设计注意事项

当设计 Bootloader 系统时，需要注意某些事项。本节将介绍部分注意事项。

5.1 Bootloader 的替代方法

如上面所述，通过 Bootloader，可以现场更新产品固件。

可以使用另一种方法解决该问题。例如，在应用程序本身内部编程闪存更新功能。但是，后面的应用程序必须要覆盖自身的部分或所有内容，这样会增加复杂性。因此，将 Bootloader 分为单独模块或程序是更好的设计实践。

5.1.1 HSSP

Bootloader 的另一个替代方法是使用主机源串行编程（HSSP）。通过该方法，外部主机可以直接控制 MCU 的 JTAG 或 SWD 引脚，从而将新的应用程序编程到闪存内。赛普拉斯的 CY8CKIT-002 MiniProg3 和其他编程器都使用了该方法。

虽然在开发程序和基于工厂的编程过程中通常使用的是 HSSP，但在现场不经常使用它。只有在带有多个 MCU 的 PCB 上才会使用 HSSP，其中一个 MCU 能直接编程另一个 MCU。

更多有关访问 PSoC JTAG/SWD 引脚的详细信息，请查阅下面文档：

- AN61290，PSoC 3 和 PSoC 5LP 硬件设计注意事项
- AN88619，PSoC 4 硬件设计注意事项
- PSoC 3 编程规范
- PSoC 4 编程规范
- PSoC 5LP 编程规范

欲了解有关 HSSP 的信息，请查阅下面文档：

- AN73054：使用外部微控制器（HSSP）对 PSoC 3 和 PSoC 5LP 进行编程
- AN84858：使用外部微控制器（HSSP）对 PSoC 4 进行编程

5.2 存储器使用情况和模块配置

如上面所述，应该将 Bootloader 代码和应用程序代码相互隔离开 — 它们通常被设计为单独的模块。由于这两个模块必须位于闪存中，那么 Bootloader 的位置在哪里？某些 MCU 包含了独立于闪存的硬编码引导加载只读存储器（ROM），如图 5 所示。其他 MCU 将闪存部分使用于 Bootloader，如图 6 所示。

图 5. 单独的 ROM 存储器中的 Bootloader

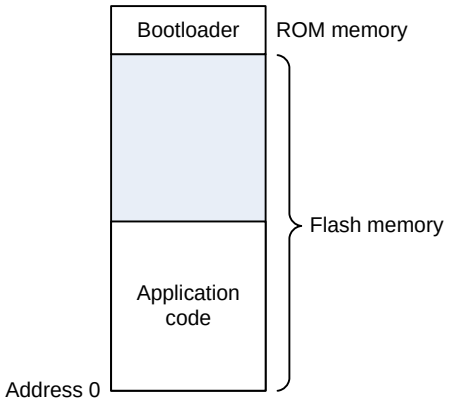
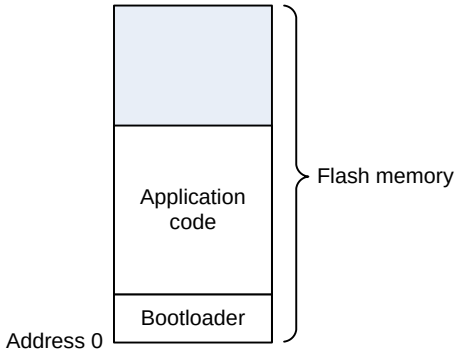


图 6. 闪存存储器中的 Bootloader



使用 ROM 方法可使应用程序使用所有闪存，而闪存方法则允许对 Bootloader 进行灵活的设计。通常优先选择闪存方法。通常将 Bootloader 放置在起始地址为 0 的闪存内。由于几乎所有 CPU 都是在地址 0 上开始执行代码，因此在器件复位时，Bootloader 先运行。

使用闪存方法可能出现的问题是：Bootloader 会使用应用程序应该使用的存储空间。如果 MCU 应用程序所需的存储空间较大，这样会影响应用程序的设计和成本。因此，当使用闪存方法时，应尽量减少 Bootloader 的容量。要想降低 Bootloader 的大小，请尽可能把各项要求和设计变得简单；请参考第 10 页上的 Bootloader 存储器的使用情况。如果必须添加其他功能，则难以保持 Bootloader 的最小容量；请参考第 8 页上的自定义程序。

5.3 Bootloader — 主机时序

Bootloader 设计中的一个重要注意事项是同主机开始通信的时序。如图 4 所示，在确定应用程序有效后，Bootloader 将等待一段时间，这样主机可以开始新的引导加载操作。

等待时间通常为 50 到 500 毫秒。如果等待时间过短，主机将无法开始通信。如果等待时间过长，则产品整个启动时间会太久。

时序问题的另一个解决方案是允许应用程序调用 Bootloader。然后，应用程序将对某些外部事件（如按下按键或从主机的信息）做出响应，并开始引导加载操作。

5.4 通信端口

在大多数情况下，根据整个产品的要求设置图 3 中通信端口的规范。除了这些要求外，为获取 Bootloader 系统的强大支持，端口应该进行以下操作：

- 基于数据包数据的传输。该端口不应该解析数据包；该任务应该由 Bootloader 和主机负责。
- 数据包错误检测。该端口能够检测和报告带有无效数据的数据包。系统的其他部分会处理无效数据包报告。
- 命令响应协议。一般情况下，主机将命令数据包发送给 Bootloader，然后等待成功/失败状态响应数据包。
- 中等速度传输。由于写入到单个闪存行可能需要几毫秒，因此使用高速端口也不会显著缩短引导加载总时间。
- 在对闪存进行写操作时仍可以执行其他传输。这样，当将前一行写入到闪存时，可以下载另一个数据行。

在嵌入式系统中的所有通用协议当中，USB 是支持这些特性的最好器件（虽然 USB 代码需要使用大量的闪存）。UART、I^2C 和 SPI 更加简单，但需要用于管理数据包的额外代码。请注意，I^2C 由主设备端（通常为主机）全权控制，这样更加难以实现命令响应协议。

5.5 从失败中恢复

Bootloader 应该能够检测、报告并处理引导加载过程中发生的错误，如电源故障、通信中断和闪存写错误。

通常通过将应用程序的某些校验位（校验和或 CRC）存储在闪存内，可实现该操作。当启动引导加载操作时，会清除这些位。如果应用程序被成功下载和安装，将更新这些位。例如，如果电源故障是在引导加载过程中发生的，那么 Bootloader 将在复位时检测无效校验位，并且不会将控制权交给局部加载的应用程序。反而，它会等待主机启动另一个引导加载操作。

5.6 将来可靠性保证

另一个设计注意事项为：在安装后不能现场更新 Bootloader。当然可以配置一个能够在现场进行自我覆盖或更新的 Bootloader，但这样做比较复杂，最好避免这样的操作。要使 Bootloader 更加强大且确保其将来的兼容可靠性，关键是尽量把各项要求和设计变得简单。要想避免缺陷，需要使用编码的最佳实践并彻底检查代码。

由于 Bootloader 和应用程序是单独的模块，因此您可以使用不同的编译器或不同的开发系统对它们进行编译。由于在不同的版本中各种工具（如编译器）会发生改变，因此需要确保两个模块之间的传输控制机制保持不变。另外，随着时间的推移您升级了开发工具时，请确保旧版本的 Bootloader 仍可以加载新的应用程序。

5.6.1 应用程序管理

由于 Bootloader 和应用程序是单独的模块，而且应用程序会发生改变，因此您必须考虑如何将控制权从 Bootloader 传输到应用程序。下面介绍的是实现该操作的一些方法：

- 跳转到应用程序始终开始的固定位置。该方法比较简单，但没有为 Bootloader 或应用程序提供足够的灵活性用于将来进行更改。
- 将应用程序的起始地址保持在闪存的公共区域内。然后，Bootloader 会将该位置作为指向应用程序起始地址的指针。
- 在通用的开发系统中，将应用程序链接至 Bootloader，以便 Bootloader 能够跳转到一个符号地址。

第二种方法是简单性和灵活性的最好结合，通常优先选择该方法。

5.6.2 闪存保护

Bootloader 能够在闪存存储器中检查自身的镜像，以确定它是否有效。如果该镜像无效，则 Bootloader 必须停止运行。不过，这样便无法再使用该产品。

确保 Bootloader 在闪存中一直有效的最好方法是使用硬件，这样可以确保固件永远不会覆盖 Bootloader。可以使用闪存写入保护电路来实现该操作，该电路能够防止意外覆盖掉 Bootloader 闪存。更多有关 PSoC 实现的信息，请参考第 8 页上的 Flash 保护。

5.7 自定义程序

将 Bootloader 设计为易修改的模块，以便可以用于不同的产品应用程序。例如，Bootloader 系统很容易能够使用不同的通信端口，甚至是多个通信端口。

另外，Bootloader 系统需要运行在高可靠性的产品中，该产品拥有以下三个主要特性：

- 在从 Bootloader 转换到应用程序期间，需要保持重要引脚的状态。当通过器件复位进行转换时，该操作会成为一个要考虑的事项。更多有关信息，请参考附录 A。
- 在进行引导加载的同时，需要执行重要的任务。需要将额外代码添加到 Bootloader 内，从而使能多任务系统。请参考第 11 页上的自定义程序。
- 多个（通常为两个）应用镜像被存储在闪存内。如果一个镜像在闪存中被损坏，Bootloader 会将控制权交给其他镜像，从而降低您的产品在故障间等待的时间（MTBF）。PSoC Creator 支持双镜像 Bootloader。

6 PSoC Bootloader — 工作原理

在上述各节中，我们已经了解了通用 Bootloader 的功能和设计注意事项。现在，我们会通过使用 PSoC Creator 集成设计环境（IDE）了解如何在 PSoC 3、PSoC 4 和 PSoC 5LP 中实现这些原理。

PSoC 器件拥有存储器和可配置的外设硬件，因此可创建功能强大且灵活的 Bootloader 系统。使用 PSoC Creator 进行开发，它是赛普拉斯提供的免费 IDE，用于编译基于 PSoC 的解决方案。有关 PSoC 器件的详细信息，请参考：

- AN54181：PSoC 3 入门
- AN79953：PSoC 4 入门
- AN77759：PSoC 5LP 入门

有关 PSoC Creator 的更多信息，请参考 PSoC Creator 主页。

注释： PSoC 3、PSoC 4 和 PSoC 5LP 中 Bootloader 系统的实现与 PSoC 1 不一样。有关 PSoC 1 Bootloader 的详细信息，请参考 AN2100：Bootloader：PSoC 1。

与赛普拉斯的所有 PSoC 产品和支持 IDE 的产品相同，PSoC Creator 通过自动实现基本的系统功能来缩短您的设计时间。Bootloader 并不例外 — 实际上它只需要几分钟就可以将一个简单的 I^2C Bootloader 添加到您的项目中。更多有关信息，请参考将 Bootloader 添加到 PSoC Creator 项目中。

6.1 PSoC Creator Bootloader 项目

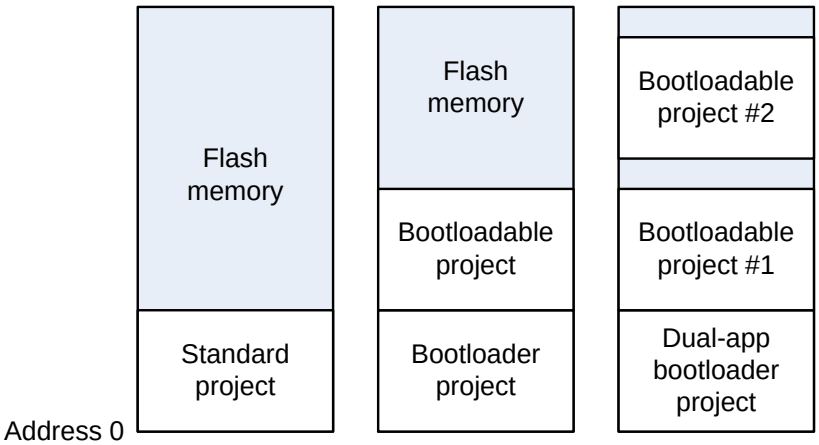
PSoC Creator 将一个完整且独立的应用程序定义为“项目”。除了源代码外，项目还带有数据字节，这样可用于为您的应用程序配置 PSoC 器件的模拟和数字外设。

请注意，当使用 PSoC Creator 时，Bootloader 和应用程序会在完全独立的项目中实现。可用的项目类型为：**Standard**（或“通用”、“无 Bootloader”）、**Bootloader** 和 **Bootloadable**。第四种项目类型，即 **Dual-App Bootloader**，支持高度可靠的应用程序的双应用程序镜像，如自定义程序中所述。您可以轻松更改项目类型，例如将 Standard 改为 Bootloadable。

您必须将 Bootloadable 项目同相应的 Bootloader 项目结合在一起。可以将 Bootloader 项目同多个 Bootloadable 项目结合在一起。

由于 PSoC 没有引导加载 ROM，因此 Bootloader 被放置在闪存内，如图 7 所示。Bootloader 项目放置在闪存内的地址 0，并且在器件复位时先被执行。这时它会实现显示在图 4 中的程序流。

图 7. PSoC Creator 项目和闪存的使用情况



6.2 Bootloader 选项

PSoC Creator 提供了 Bootloader 组件，该组件拥有各种配置选项，用于设置 Bootloader 的运行时特性。各种选项如下所示：

- 等待命令：在将控制权传递给 Bootloadable 前是否要等待来自主机的命令。
- 等待命令时间：超时为 1 到 2550 毫秒，或永远等待。仅在“等待命令”为“是”时有效。
- 通信组件：Bootloader 使用的通信组件。PSoC Creator Bootloader 支持多种通信端口类型，包括一个自定义选项。
- 校验和类型：用于检查发送到主机或从主机接收到的数据包的校验位：“校验和”或“CRC”。

更多有关信息，请参考第 14 页上的图 11 以及 Bootloader 组件数据手册中介绍的内容。

6.3 通信组件

PSoC Creator Bootloader 项目必须至少包含一个与 Bootloader 相兼容的通信组件。目前，I2C 从设备、UART、SPI 和 USBFS 组件是支持 Bootloader 的通信标准。

如果您想将非标准的通信通道用于引导加载，那么很容易便能够创建一个自定义组件。您必须为该组件编写一个仅支持五种功能（启动、停止、复位、读取和写入）的 API。有关如何创建 Bootloader 自定义通信组件的更多信息，请参考 PSoC Creator 组件创建指南中的内容。

6.4 从故障恢复

PSoC Creator Bootloader 组件使用闪存最上面一行（128 或 256 字节）来存储应用程序（或双应用的两个应用程序）中的数据。该数据包括了每个应用程序的校验和以及其他有效位。当启动引导加载时，这些位会被清除。引导加载成功完成后，这些位被重新计算和更新。

如果在引导加载过程中发生电源故障或通信中断，则 Bootloadable 项目的校验和会在下一次器件复位中为错误。Bootloader 在此将等待主机发出另一个命令，从而启动另一个引导加载操作。

6.5 向后兼容

PSoC Creator 设计保证新版本的 Bootloadable 项目能够与旧版本的 Bootloader 项目相链接和相兼容。

6.6 Bootloader 存储器的使用情况

如上面所述，PSoC Creator Bootloader 使用了应用程序应该使用的存储器。这样会影响应用程序的设计或成本，因此 Bootloader 存储器的使用情况是一项非常重要的规范。PSoC Creator Bootloader 组件的存储器使用情况明显不同样，它取决于：

- 所用的通信组件
- 所选的 Bootloader 组件配置选项（请参考第 14 页上的图 11）
- 目标器件 — PSoC 3、PSoC 4 和 PSoC 5LP 分别拥有 8051、Cortex-M0 和 Cortex-M3 CPU
- 编译器和其优化设置

更多有关信息，请参考 Bootloader 组件数据手册中列出的规范。对于特殊的 Bootloader 项目，在编译项目后，请检查编译器生成的.*map* 文件，用以确定存储器的准确使用情况。

6.7 Flash 保护

所有 PSoC 3、PSoC 4 和 PSoC 5LP 器件均包含了控制对闪存的访问的灵活闪存保护系统。该性能不但可以保护专有代码，并且还可以防止对闪存内 Bootloader 部分执行写操作。

闪存以行的形式构成，根据具体的产品系列，每一行的大小范围为 64 到 256 字节。您可以对每一行分配四个保护级别的其中一个（PSoC 4 具有两个保护级别）；请参考表 1。 要想更改闪存保护级别，必须擦除整个闪存。有关 PSoC 闪存和安全特性的更多信息，请参考器件数据手册或技术参考手册（TRM）。

表 1. 闪存保护级别

	PSoC 3 和 PSoC 5LP		PSoC 4	
保护设置	**支持**	**不支持**	**支持**	**不支持**
无保护	外部读写操作， 内部读写操作	–	外部读写操作， 内部读写操作	–
出厂升级	外部写操作， 内部读写操作	外部读操作	NA	NA
现场升级	内部读写操作	外部读写操作	NA	NA
全面保护	内部读操作	外部读写操作， 内部写操作	内部读操作	外部写操作， 内部写操作 （请参考下面的注意事项）

注释： 为了防止对 PSoC 4 器件进行的外部读操作，您必须在 PSoC Creator *.cydwr* 系统设置中将器件保护设置更改为“受保护”，并使用 PSoC Programmer 软件来编程器件。编程器件前，还必须使能 **Options** > **Programmer Options** 中的 “Chip Lock”（芯片锁定），这样能够使设置生效。

通过将相应行设置为“全面保护”，可以保护闪存的 Bootloader 部分。PSoC Creator 有助于您容易选择每一个行的保护设置。更多有关信息，请参考 PSoC Creator 帮助参考或相关应用笔记中列出的高级 Bootloader 应用笔记。

6.8 自定义程序

一个 Bootloader 便是一个 PSoC Creator 项目；与任何其他 PSoC Creator 项目相同，它会将 PSoC 配置为任意的应用。这样又反过来使 Bootloader 易于制定，尤其是对于高度可靠的应用：

- 引导加载过程中其他任务：可以将组件添加到 Bootloader 项目原理图中；在多种情况下，这些组件无需使用 CPU 仍能执行复杂的任务。

  如果在引导加载过程中需要使用 CPU 来执行其他任务，最简单的方法是将任务构成一个状态机，并将其嵌入到周期性中断处理程序中。使用这种方法，Bootloader 和次要任务可以独立运行。

- 预留引脚状态：可将引脚组件放置在原理图中，并将它们的状态设置为器件复位和 Bootloader 启动。有关控制引脚状态的更多信息，请参考 AN61290：PSoC 3 和 PSoC 5LP 硬件设计注意事项或 AN88619：PSoC 4 硬件设计注意事项。另外，请参考附录 A。

7 将 Bootloader 添加到 PSoC Creator 项目中

我们已经了解了如何在 PSoC Creator 中实现 Bootloader，那么现在需要了解执行该操作的一些实际步骤。更多有关信息，请参考相关应用笔记中列出的高级 Bootloader 应用笔记。

7.1 建立一个 Bootloader

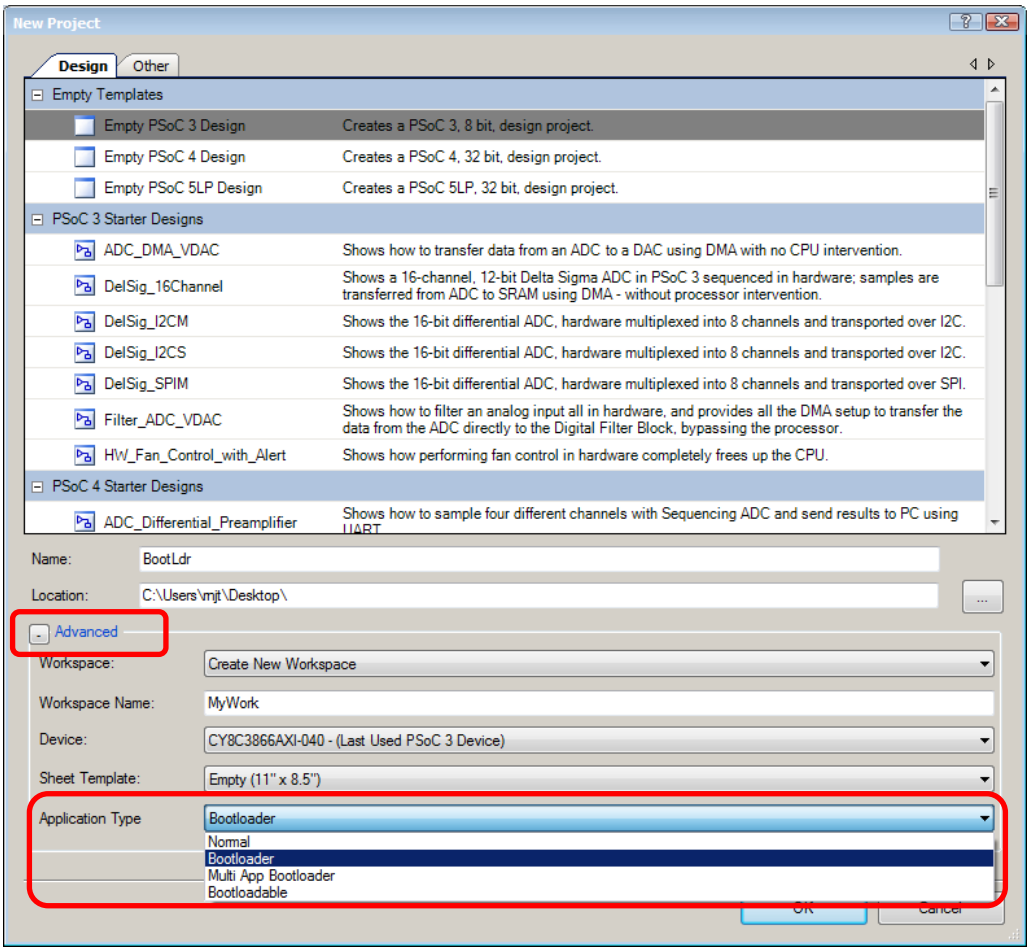
对于其他 MCU，您通常要将 Bootloader 添加到某个应用中。但对于 PSoC Creator，最好的方法是进行相反的设计，即为：先创建一个 Bootloader 项目，然后再创建一个或多个 Bootloadable 项目。

要创建一个 Bootloader 项目，简单创建一个类型为 **Bootloader** 或 **Multi-App Bootloader** 的项目，如图 8 所示。请注意，在该示例中名为“BootLdr”的项目与名为“MyWork”的工作区不一样 — “MyWork”工作区即是可包含多种类型的项目的 PSoC Creator 工作区。

注释： 若从 **PSoC Creator 3.2** 开始，在 New Project 对话框中删除了“Application Type”选项。PSoC Creator 自动识别 TopDesign 原理图中的应用类型。然而，该更新不会影响现有的 Bootloader 项目。

注释： 若从 PSoC Creator 3.3 开始，New Project 对话框发生了改变，您不需要选择“Application Type”选项。

图 8. 在 PSoC Creator 3.1 或更早版本中创建 Bootloader 项目



创建一个项目后，将 Bootloader 组件和用于引导加载的通信组件添加到项目原理图中。如图 9 所示，Bootloader 和 Bootloadable 组件会显示在 Component Catalog 窗口中的 System 选项卡下。图 10 显示了具有一个 Bootloader 组件和一个用于引导加载的 I^2C 通信组件的 Bootloader 项目原理图。

图 9. Component Catalog 窗口

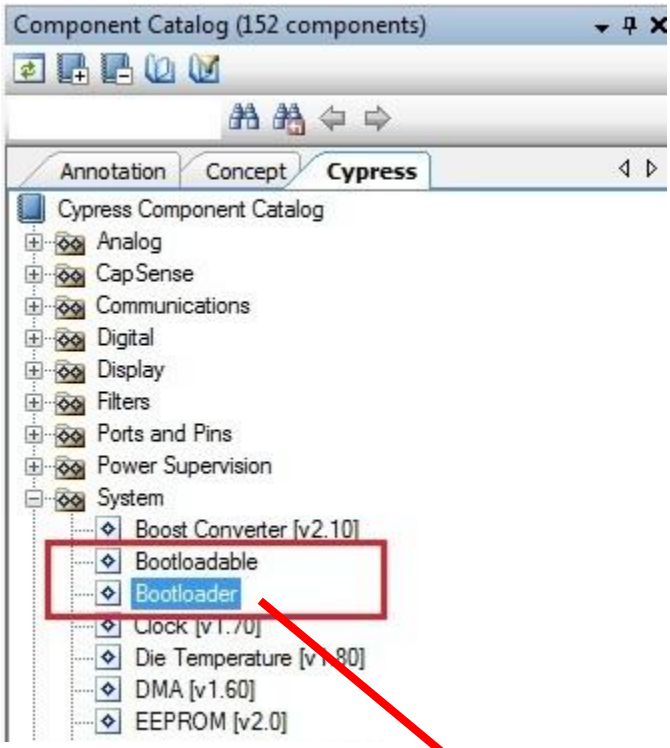
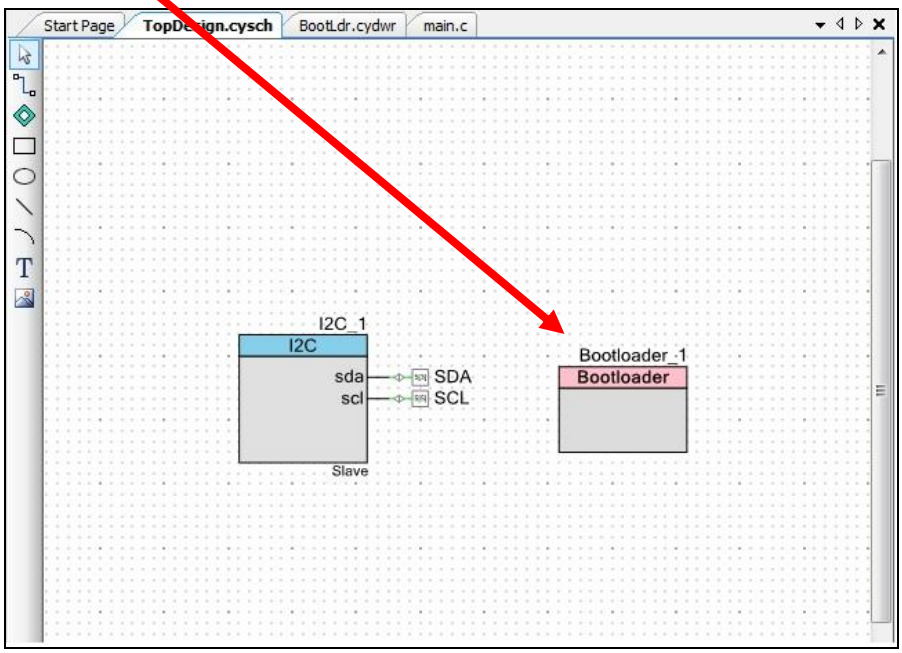
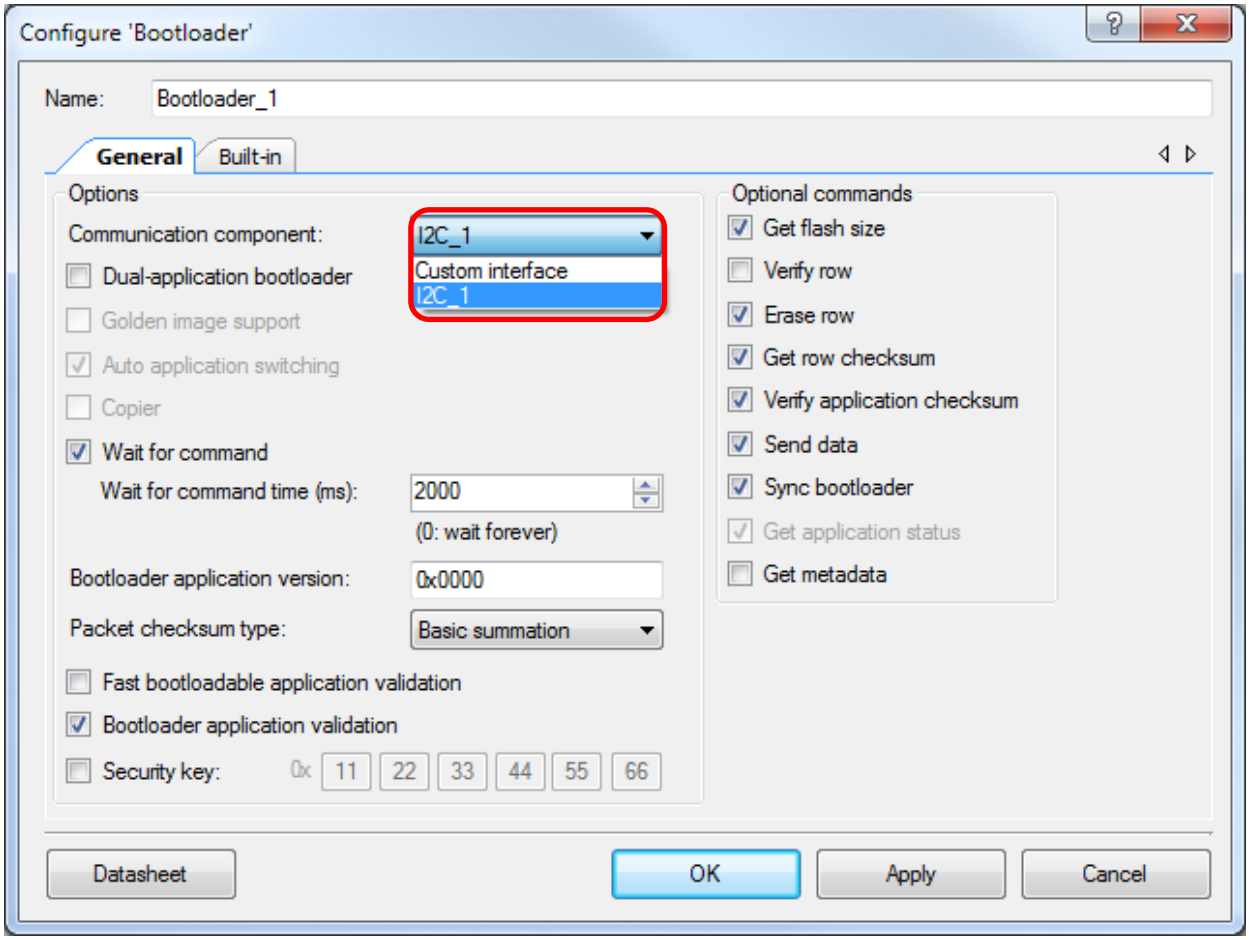


图 10. Bootloader 项目原理图



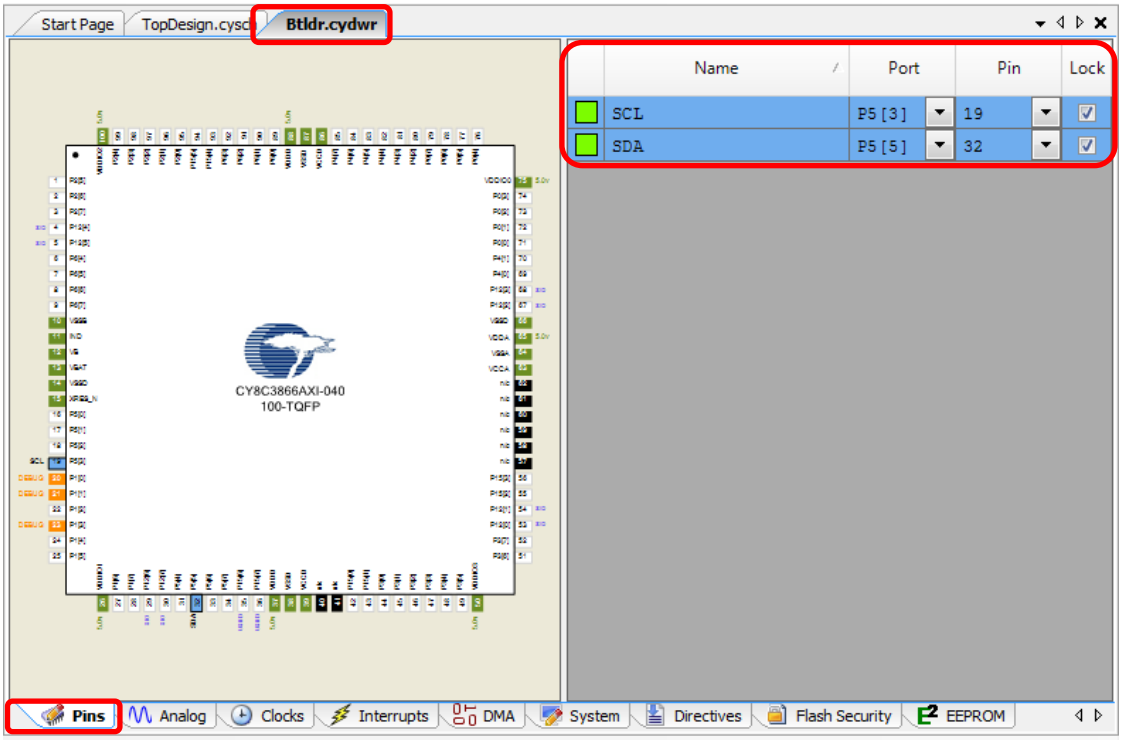
然后，配置 Bootloader 组件，如图 11 所示。请注意图 11 中用于选择通信组件的菜单 — 这是用于与主机进行通信的组件。创建 Bootloader 项目时，必须定义和选择该组件。您可以从原理图上选择某个与 Bootloader 兼容的通信组件，或者选择 **Custom_Interface** 并定义自己的通信组件。

图 11. Bootloader 组件配置



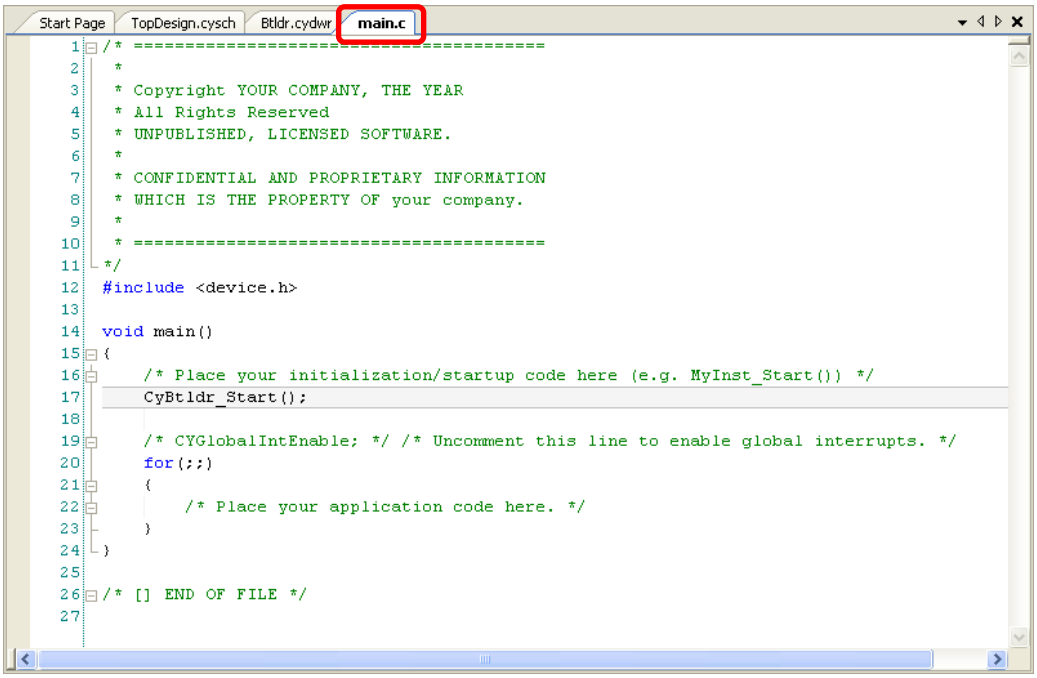
最后，在设计范围资源（DWR）窗口中，将原理图上的引脚同物理引脚连接起来，从而完成项目，如图 12 所示。

图 12. Bootloader 项目引脚分配



编译项目。其余操作均自动完成，这样您便能得到一个基本的 Bootloader 项目。自动生成的 *main.c* 文件只有一个代码行，用于调用 Bootloader 启动函数，如图 13 所示。

图 13. Bootloader 的默认 *main.c*

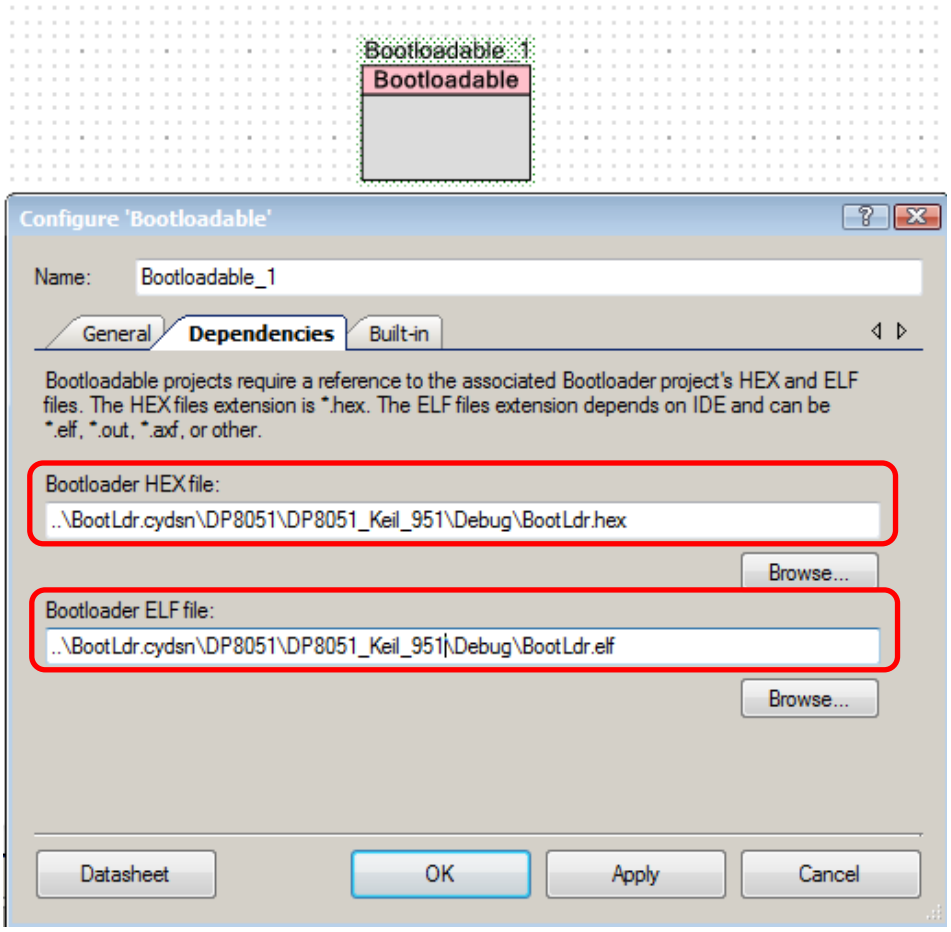


7.2 添加 Bootloadable 应用

创建 Bootloader 后，您可以根据要求通过使用第 12 页上图 8 中的 Bootloadable 选项定义多个 **Bootloadable** 应用（项目）。另外，您还可以将现有的 **Normal** 型项目更改为 **Bootloadable** 项目；有关详细信息，请参见第 17 页。

Bootloadable 项目的原理图上必须存在一个 Bootloadable 组件（请参考第 13 页上的图 9）。该项目必须与某个 Bootloader 项目相联结起来，如图 14 所示。要执行上述操作，请在 Bootloadable Component Configuration 对话框中选择 Bootloader 的.*hex* 和.*elf* 文件所在的位置；更多有关信息，请参考第 19 页上的项目文件一节。

图 14. Bootloadable/Bootloader 项目链接



一个 PSoC Creator 工作区可以具有多个项目。在多种情况下，一个 Bootloader 项目和与其相关的 Bootloadable 项目全部位于一个工作区内。然而，Bootloader 和 Bootloadable 项目也可以位于 PC 上单独的工作区中和单独的位置上。开始使用 PSoC 前，最好对整个系统的开发需求做出一个工作区/项目计划。

注释： Bootloadable 项目的闪存保护设置被忽略；相关 Bootloader 项目的闪存保护设置得到优先。

注释： 对于 PSoC Creator 3.0 以前的版本，如果 Bootloader 被更新，您必须根据 Bootloader 项目重新建立所有相关 Bootloadable 项目。使用“Clean and Build”选项。

7.3 调试 Bootloadable 项目

在 PSoC Creator Bootloader 系统中，先执行 Bootloader 项目，然后再执行 Bootloadable 项目。通过软件控制的器件复位，可以从 Bootloader 转为 Bootloadable 项目；详细信息，请参考附录 A。该操作会复位调试器接口，这样 Bootloadable 项目便不能在调试器模式下运行。

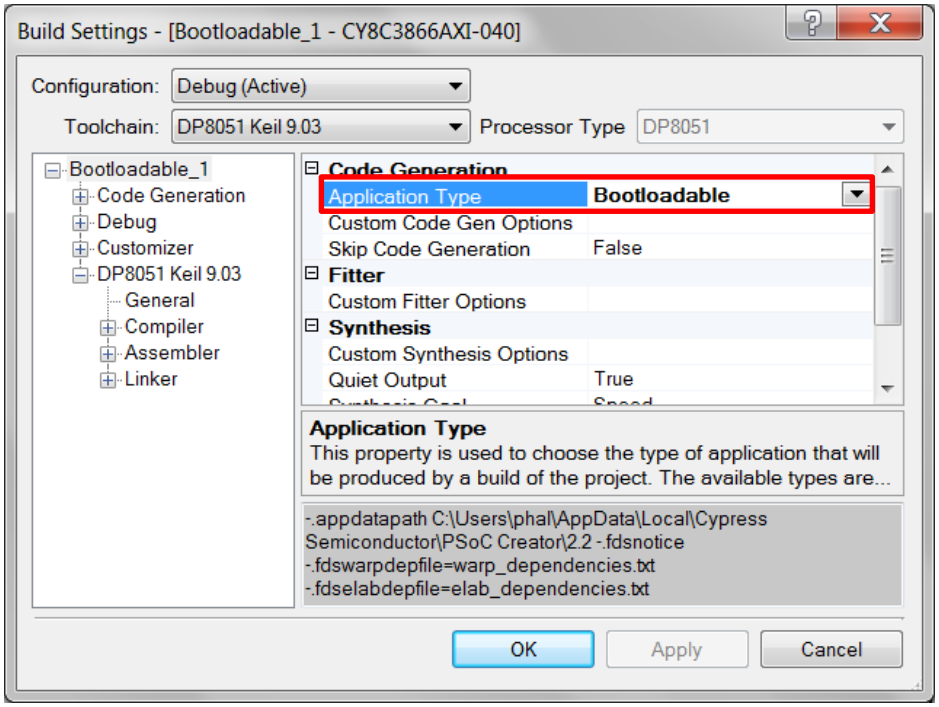
要想调试某个 Bootloadable 项目，首先将它的应用类型改为“Normal”（图 15），然后进行调试，最后再将它转换为 Bootloadable 类型。

另一个选择是在将 Bootloadable 项目.*hex* 文件编程到器件内后，在执行 Bootloadable 项目期间，使用 **Attach to running target** 选项进行调试。在这种情况下，仅在将调试器添加到器件上时，才能开始调试 Bootloadable 项目。

7.3.1 将应用项目类型从 Normal 转换为 Bootloadable

如果您已经创建了某个标准（Normal）项目，并想将该项目转换为 Bootloadable 项目，那么，请将其 **Application Type** 改为 Bootloadable，如图 15 所示。

图 15. 将应用类型改为 Bootloadable



修改应用类型后，您必须将 Bootloadable 组件添加到项目原理图中，并添加 Bootloader 项目的.*hex* 文件作为一个依赖属性，如图 14 所示。

注释： 若从 PSoC Creator 3.2 开始，在 Build Settings 对话框中删除了“Application Type”选项。PSoC Creator 自动通过 TopDesign 原理图中的组件识别应用类型。如果 TopDesign 原理图中具有一个 Bootloadable 组件，PSoC Creator 会判断项目是 Bootloadable 类型。

7.4 自定义 Bootloader

如第 8 页上的自定义程序所述，您可以通过将额外组件拖放到原理图中并将代码添加到 *main.c* 内，从而实现自定义您的 Bootloader 项目。简单的示例是再添加一个 PWM、一个时钟和一个引脚组件来高亮一个作为“引导加载”指示器的 LED，如图 16 和图 17 所示。您可以通过简单的组件配置来使 LED 在任意频率和占空比下闪烁。

请注意，PSoC 的 Bootloader 项目配置仅在 Bootloader 把控制权交给 Bootloadable 之前有效。然后，PSoC 器件将重新配置为 Bootloadable 项目。如果您想让这两个项目执行同样的功能，可以将相同的组件和代码放置在这两个项目中。

图 16. Bootloader 项目自定义程序

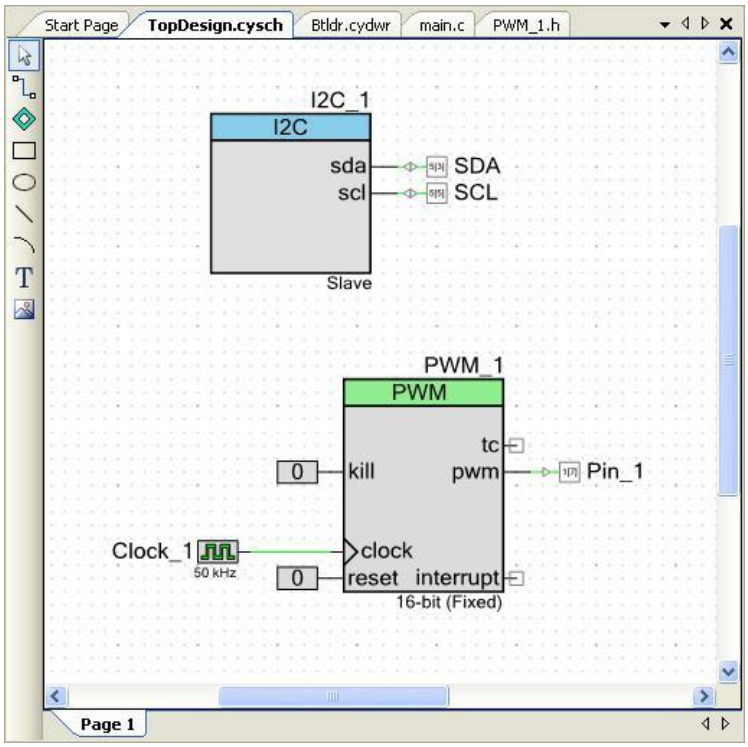
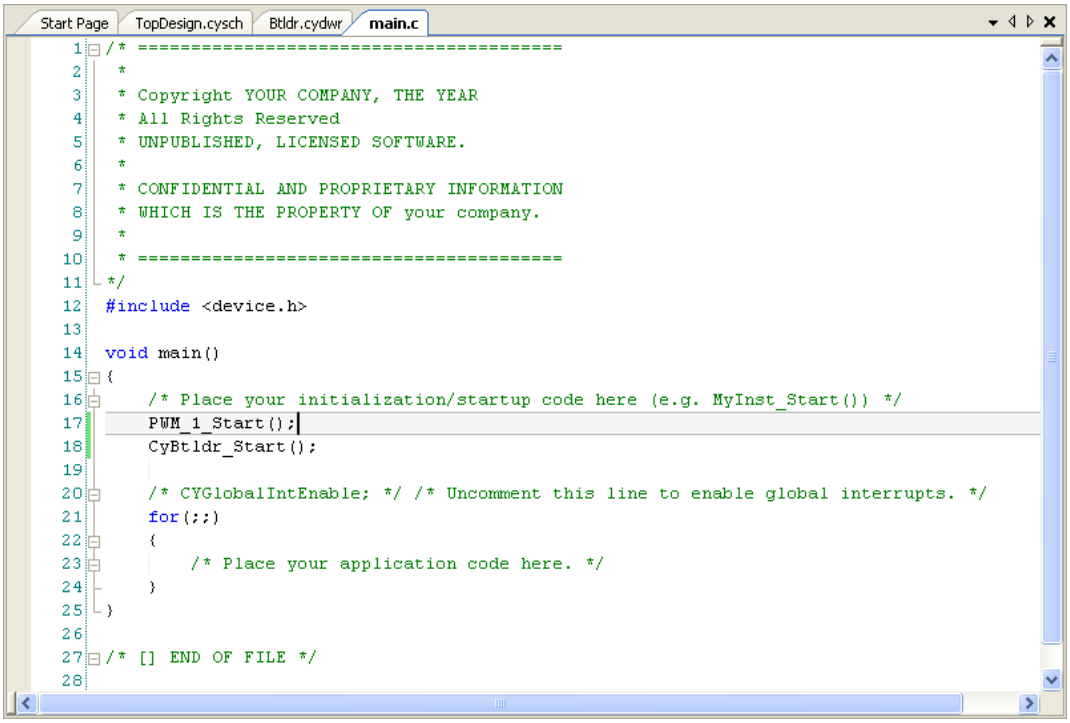


图 17.*main.c* 中的 Bootloader 自定义程序



7.5 调用 Bootloader

如 Bootloader — 主机时序中的介绍，通过使用应用来调用 Bootloader，可以避免 Bootloader 的初始时序问题。然后，应用程序将响应某些外部事件（如按下按键或来自主机的信息），并开始引导加载操作。

Bootloader 组件拥有一个带公共函数 Bootloader_Start()的应用编程接口（API）。从 Bootloadable 项目代码中调用该函数，来启动引导加载操作。

Bootloader_Start()会对器件执行软件复位，并且 Bootloader 会控制 CPU。对 Bootloader 重新配置资源和外设；禁用 Bootloadable 的配置。不会执行 Bootloadable 项目代码（包括中断处理程序）。完成引导加载操作后，会再次复位 CPU。更多有关信息，请参考附录 A。

8 将您的项目加载到 PSoC 内

与标准项目相同，通过 PSoC Creator 或 PSoC Programmer 可以在目标 PSoC 器件中使用 JTAG 或 SWD 接口来安装 Bootloader 项目。Bootloader 被安装并生效后，通过引导加载操作（而不是通过 JTAG 或 SWD）来安装 Bootloadable 项目。

PSoC Creator 包含一个名称为 Bootloader 主机的 PC 程序，用于执行 PC 的引导加载操作。通过使用一个 Programmer（如 CY8CKIT-002 MiniProg3 套件），它能够直接通过 JSB 端口或 I^2C 端口与目标 PSoC 中的 Bootloader 通信。要想使用 Bootloader 主机，需要了解为 Bootloader 和 Bootloadable 项目生成的输出文件。

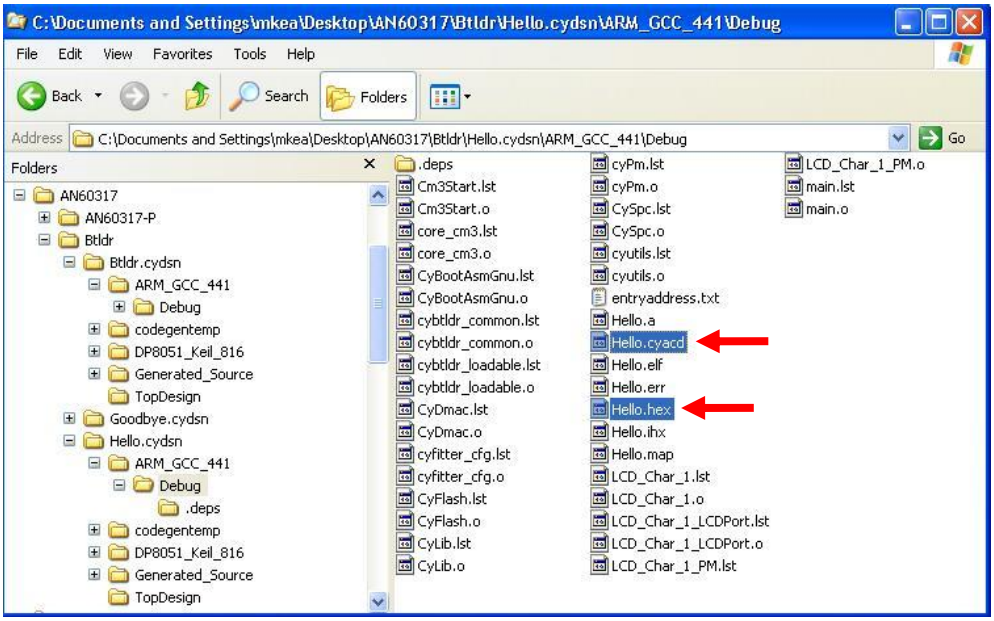
8.1 项目文件

一旦进行了编译，所有 PSoC Creator 项目都会生成*.hex* 文件作为输出内容。这些文件可以适用于所有 HSSP Programmer 器件（包括 MiniProg3）。十六进制文件包含了 CPU 代码和项目配置的多个字节。

对于 Normal 型和 Bootloader 型项目，*.hex* 文件包含了只适用于该项目的代码和数据字节。Bootloadable *.hex* 文件则不一样，它包含 Bootloadable 项目*和*相关 Bootloader 项目的代码和数据字节 — 两种项目同时进行编程。

Bootloadable 项目也同 Normal（正常）型项目不一样，它会生成第二个文件（类型为*.cyacd*）作为一个输出，如图 18 所示。*.cyacd* 文件包含的代码和数据只用于 Bootloadable 项目，不用于相关的 Bootloader 项目。这样是为了在 Bootloader 主机程序中使用该文件，并将该文件下载到目标 PSoC 内；该 PSoC 具有已经安装的相关 Bootloader 项目。

图 18. Bootloadable 项目文件



8.2 使用情况

编译项目和创建输出文件后，通常使用下面一种方案（请参见第 5 页上的图 4）：

1. 创建和编译一个 Bootloader 项目。通过使用一个 HSSP Programmer（如 MiniProg3）将其.*hex* 文件编程到目标 PSoC 内。
2. 复位目标 PSoC，以启动 Bootloader。因为 Bootloader 是闪存中唯一的项目，所以它会永远等待主机发出的引导加载命令。
3. 创建一个 Bootloadable 项目（它与 Bootloader 项目相关联），并编译它。使用主机程序和以前安装的 Bootloader，将它的.*cyacd* 文件下载到目标器件内。

对于后续的引导加载操作需要多加注意，因为有效的 Bootloadable 项目位于闪存内，Bootloader 仅在很短的时间内等待主机发出命令，然后将把控制权交给 Bootloadable。

在工厂生产过程中，您可以执行下面方案：

1. 创建和编译一个 Bootloader 项目。
2. 创建一个 Bootloadable 项目（它与 Bootloader 项目相关联），并编译它。通过使用一个 HSSP Programmer（如 MiniProg3）将它的.*hex* 文件（包含 Bootloader 和 Bootloadable）编程到目标 PSoC 器件内。
3. 复位目标 PSoC 器件，以启动 Bootloader。Bootloader 发现闪存中有效的 Bootloadable，并经过等待主机发出引导加载命令的超时后，它会把控制权交给 Bootloadable。

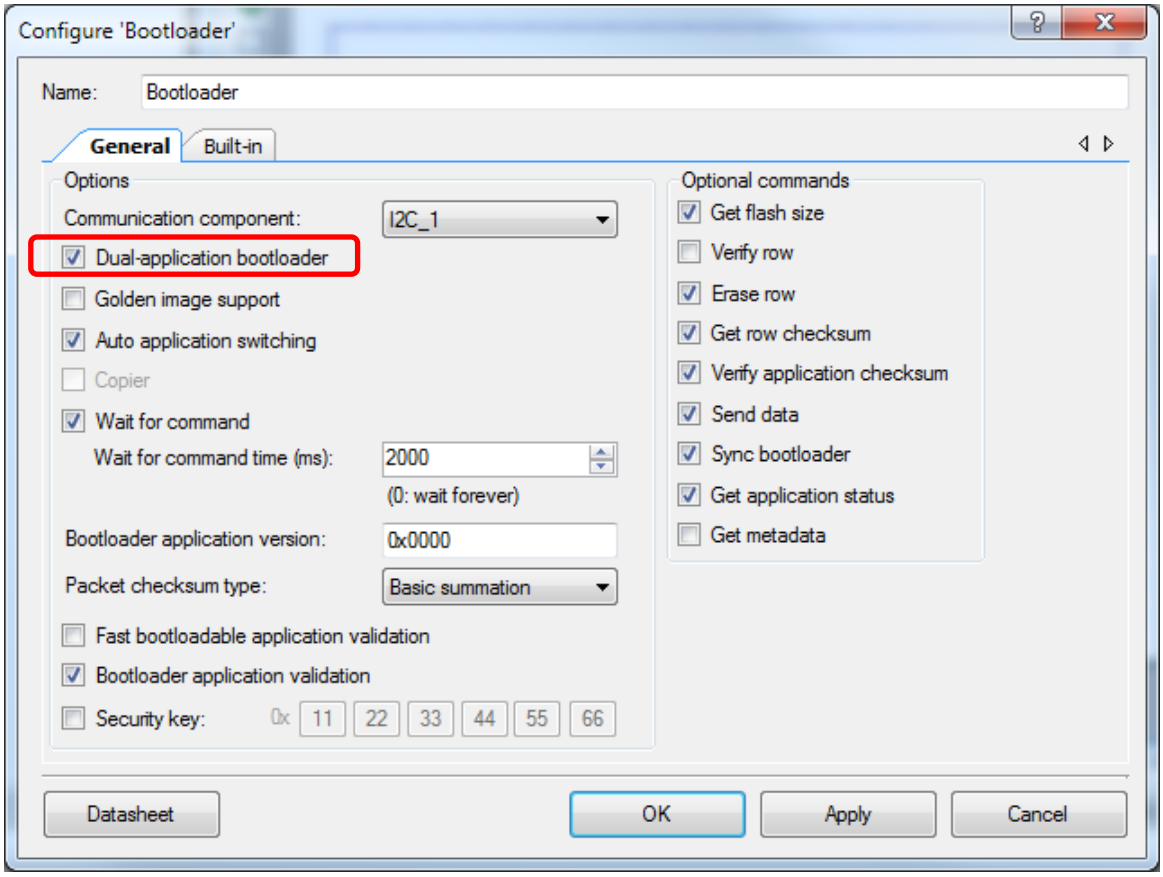
如果以后更新了 Bootloadable，您通过使用主机程序可以将.*cyacd* 文件下载到目标器件内。这样会覆盖掉以前版本的 Bootloadable。

9 双应用 Bootloader 的注意事项

PSoC Creator Bootloader 和 Bootloadable 组件为高可靠性应用程序支持双应用镜像，如自定义程序中所述。PSoC Creator 对双应用 Bootloader 的编译过程与单应用的相似，但仍存在以下差异：

1. 勾选 **Multiple-application bootloader**（请参考图 8），以创建一个双应用 Bootloader 项目。对于 PSoC Creator 3.2 和更高版本，不需要进行该步骤。
2. 在 Bootloader 组件配置对话框中，勾选 **Dual-application bootloader** 复选框，如图 19 所示。在 PSoC Creator 3.2 和更低版本，该选项的名称为 **Multi-Application bootloader**。

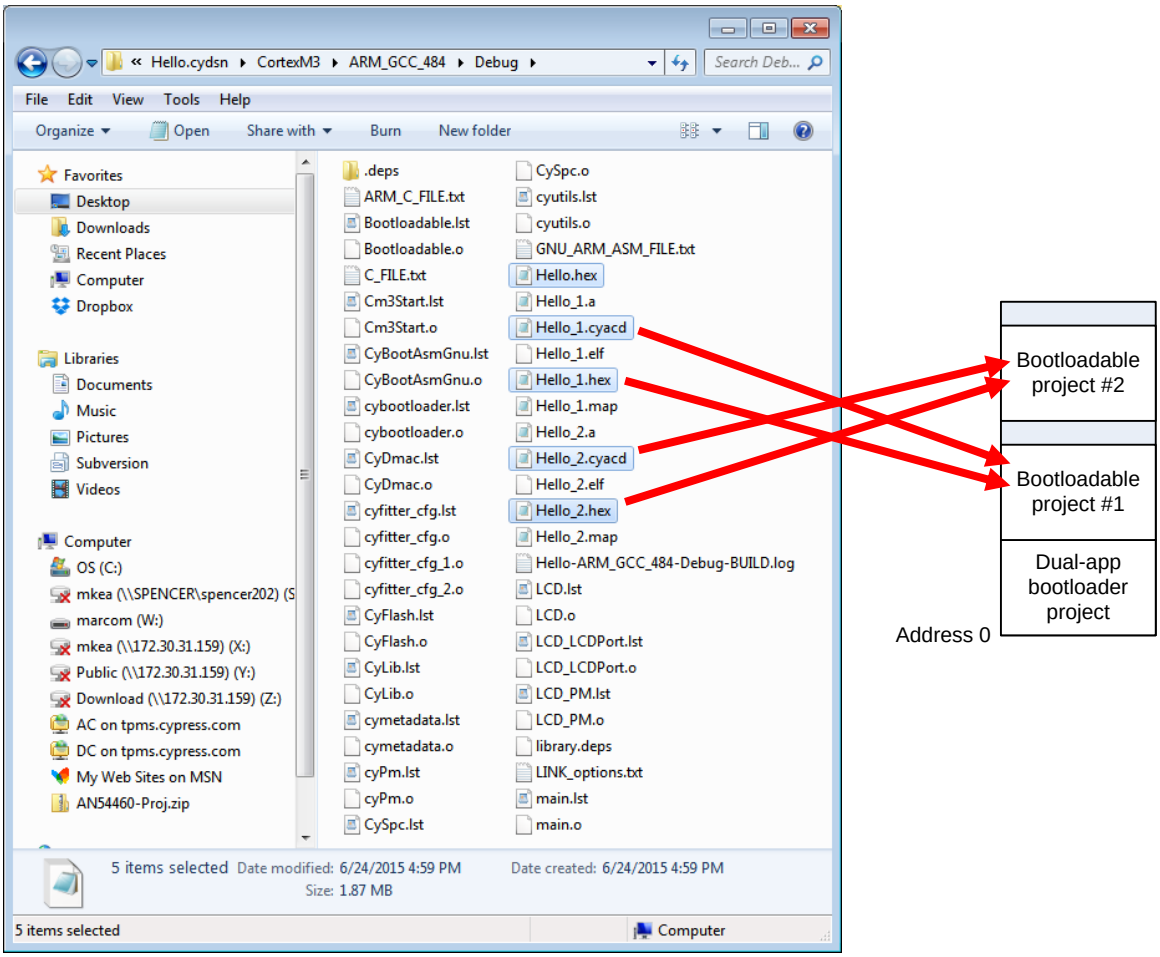
图 19. Bootloader 组件配置



3. 项目文件：双应用 Bootloadable 项目具有 5 个输出文件，而不是如图 18 中所示的两个文件。这些文件允许将 Bootloadable 项目选择为应用 1 或应用 2，如图 20 所示。对于高可靠性的应用，您可以将相同 Bootloadable 项目的两份镜像文件保存在闪存中；

   也可以创建两个不同的 Bootloadable 项目。然后可以将其中一个项目作为第一个应用，另一个项目作为第二个应用，并进行安装。

图 20. 双应用 Bootloadable 项目文件



如前面所述，*.hex* 文件通常在制造产品时通过 JTAG/SWD 端口进行安装。*.hex* 文件包含了 Bootloader 项目和一个/两个 Bootloadable 项目。

.cyacd 文件都是在产品出厂后通过 Bootloader 主机程序安装的。

9.1 应用启动流程

双应用 Bootloader 必须决定需要“启动”的应用（若有），或将控制权交给哪个应用。每个应用均具有以下两个特性来影响该决定：

- 使能（Active）：如前部分所述，PSoC Creator Bootloader 组件使用闪存的顶部行来存储应用中的数据（被称为“metadata”（元数据））。这些数据包含一个“使能”位。只有一个应用的使能位得到设置，这样该应用也是优先启动的应用。
- 有效（Valid）：启动应用前，Bootloader 会测试每个应用中元数据内的检测字节，从而确定有效的应用（若有）。由于闪存存储器受损坏或闪存中没有安装任意应用，因此该测试操作可能失败。此时，应用变为“无效”状态。

表 2 显示的是决定矩阵，Bootloader 使用它来指出需要启动的应用。请注意某些情况（例如：两个应用都有效）是非法的，在正常条件下不应该发生这些情况。

表 2. 应用启动的决定矩阵

	应用#1		应用#2		
案例	使能位	有效状态	使能位	有效状态	Bootloader 操作
0	0	0	0	0	停留在 Bootloader，一直等待主机
1	0	0	0	1	与案例#0 相同
2	0	0	1	0	与案例#0 相同
3	0	0	1	1	启动应用#2
4	0	1	0	0	与案例#0 相同
5	0	1	0	1	与案例#0 相同
6	0	1	1	0	与案例#0 相同
7	0	1	1	1	启动应用#2
8	1	0	0	0	与案例#0 相同
9	1	0	0	1	与案例#0 相同
10	1	0	1	0	与案例#0 相同
11	1	0	1	1	启动应用#2
12	1	1	0	0	启动应用#1
13	1	1	0	1	启动应用#1
14	1	1	1	0	启动应用#1
15	1	1	1	1	启动应用#1

10 总结

本应用笔记已经提供了各个 Bootloader 的基本概况 — 如何使用它们以及设计中重要的注意事项。另外，它还显示了 PSoC Creator 设计环境如何解决 PSoC 3、PSoC 4 以及 PSoC 5LP 器件等注意事项。

您还能了解有关如何使用 PSoC Creator 来快速、轻松地将一个 Bootloader 添加到您的设计中的基本概况。更多有关这些主题的应用笔记，请参考相关的应用笔记。

11 相关应用笔记

Bootloader 应用笔记

下面列出的所有 Bootloader 应用笔记在赛普拉斯网页上都有相关代码示例。

- AN60317：PSoC 3 和 PSoC 5LP 的 I^2C Bootloader
- AN86526：PSoC 4 的 I2C Bootloader
- AN73503：PSoC 3 和 PSoC 5LP 的 USB HID Bootloader
- AN68272：PSoC 3、PSoC 4 以及 PSoC 5LP 的 UART Bootloader
- AN84401：PSoC 3 以及 PSoC 5LP 的 SPI Bootloader

其他相关应用笔记

- AN73054：使用外部微控制器（HSSP）对 PSoC 3 和 PSoC 5LP 进行编程
- AN84858：使用外部微控制器对 PSoC 4 进行编程
- AN61290：PSoC 3 和 PSoC 5LP 硬件设计中注意事项
- AN54181：PSoC 3 入门
- AN79953：PSoC 4 入门
- AN77759：PSoC 5LP 入门
- AN2100： PSoC 1 Bootloader

关于作者

姓名：	Mark Ainsworth
职务：	首席应用工程师
背景：	Mark 获得了雪城大学计算机工程学士学位以及华盛顿大学电子工程硕士学位，并拥有多年的设计和建设嵌入式系统经验。

A 附录 A — Bootloader 和器件复位

如本应用笔记中另外指出的内容，从 Bootloader 交给 Bootloadable 控制权（反之亦然）始终要通过器件复位来执行。如果您的系统必须继续执行关键的任务，同时要将一个程序更改为另一个，则需要考虑这项内容。本节详细说明了使用复位功能的原因，以及在您的应用中复位对器件性能产生的影响。

A.1 为什么需要执行器件复位？

要想了解需要进行器件复位的原因，需要注意在您的系统中 Bootloader 和 Bootloadable 项目是完全独立的 PSoC Creator 项目。每个项目都有自己的器件配置内容。因此，将一个项目更改为另一个项目时，您可以完全重新定义 PSoC 器件的硬件功能。

为了实现复杂的自定义功能，器件配置可能包含对数千个 PSoC 寄存器的设置。对于 PSoC 的数字和模拟路由功能会发生这种情况。配置寄存器和路由时，除了要设置新配置的位之外，您还要对旧配置的位进行复位。否则，不会执行新的配置，甚至会损坏器件。

因此，对 Bootloader 项目和 Bootloadable 项目互相进行切换时，需要执行器件软件复位（SRES）操作。这样会使所有 PSoC 寄存器被复位为默认状态。然后可以开始对新项目进行配置。请注意，假设所有 PSoC 寄存器均被初始化，从而将它们复位为器件默认状态，那么配置时间和闪存使用量都被缩小。

A.2 对器件 I/O 引脚的影响

如应用笔记 AN61290：PSoC 3 和 PSoC 5LP 硬件设计中的注意事项和 AN60616：PSoC 3 和 PSoC 5LP 启动过程中的介绍，在复位和启动过程中 PSoC 的 I/O 引脚处于三种不同的驱动模式，如表 3 所示。

表 3. 器件复位时 PSoC I/O 引脚驱动模式

启动事件	I/O 引脚驱动模式	持续时间（典型值）		备注
		低速 IMO（12 MHz）	快速 IMO（48 MHz）	
器件复位（SRES）被使能 器件复位被禁用	高阻模拟	40 µs		复位被使能时，将 I/O 保持为高阻模拟模式。
非易失性锁存器（NVL）的状态被复制到 I/O 端口上 开始执行代码	NVL 设置： 高阻模拟、 上拉或下拉	~12 ms	~4 ms	持续时间取决于代码执行的速度和配置的复杂性。
配置 I/O 端口和引脚	PSoC Creator 项目配置	N/A		可能有 8 种驱动模式。有关详细信息，请参考器件数据手册。
代码达到 main()函数	代码可以更改 I/O 引脚功能	N/A		

有关 PSoC 中 NVL 的使用情况，请参考器件数据手册。在您的 PSoC Creator 项目中，NVL 设置是在两个位置中进行的：

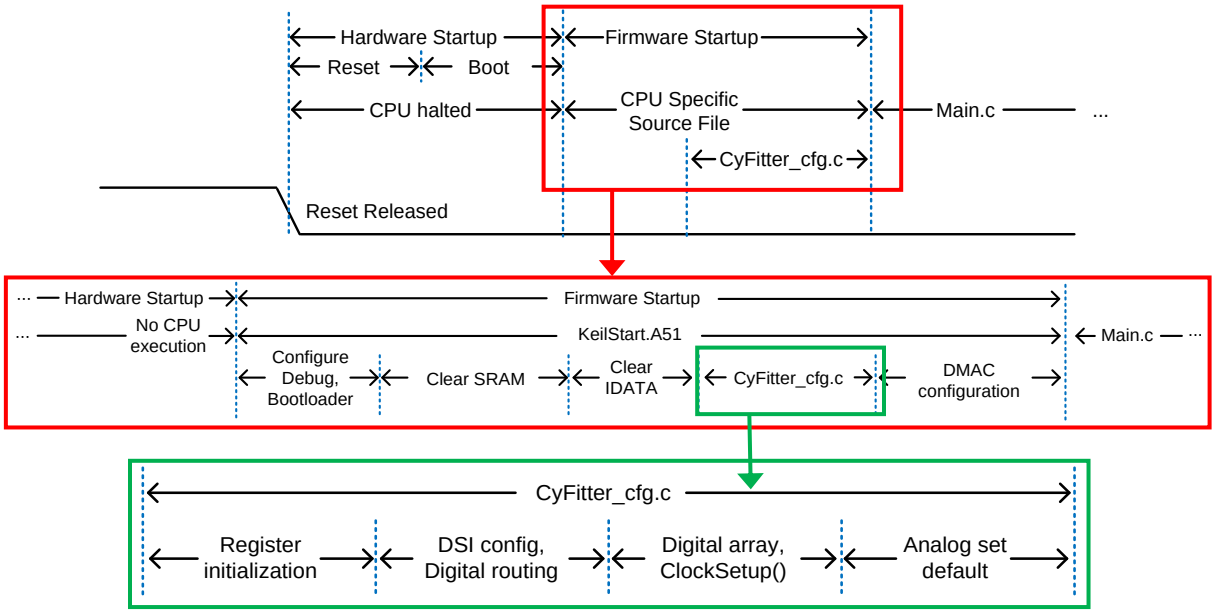
- I/O 端口：单独引脚组件配置中的 **Reset** 选项卡
- 所有其他 NVL：设计范围资源（DWR）窗口中的 **System** 选项卡

当使用您的项目来编程器件时，NVL 被更新。请注意。Bootloadable 项目不可设置 NVL；其 DWR 设置必须与相关 Bootloader 项目中的 DWR 设置相匹配。

最后 I/O 驱动模式通过单独引脚组件配置进行设置。

图 21 显示的是器件启动和配置的时序图。中间图片是 PSoC 3 的示例；PSoC 4 和 PSoC 5LP 的流程与其相似。有关详细信息，请参见 AN60616：PSoC 3 和 PSoC 5LP 启动过程。

图 21. 器件启动流程图



A.3 对其他功能的影响

器件复位时，UDB 寄存器被复位，因此不存在所有基于 UDB 的组件，并且它们的功能均被禁用。对于基于 PSoC 3 和 PSoC 5LP 中可配置的 SC/CT 模块的模拟组件也存在相同的现象（存在，但它们的功能失效）。

所有固定的外设（数字和模拟）都被复位为它们的闲置状态。这些固定外设包括 DMA、DFB、定时器（TCPWM）、I^2C、USB、CAN、ADC、DAC、比较器以及运算放大器。除 IMO 外，所有其他时钟均被停止。

所有数字和模拟路由控制寄存器都被复位。这样会打开所有数字和模拟开关，使器件的所有连接断开（包括与 I/O 相关的全部连接（与 NVL 的连接除外））。

配置后，所有硬件功能都得到恢复（请参考图 21）。开始执行项目的 main()函数时，所有固件功能都得到恢复。

A.4 示例：风扇控制

通过该示例，我们可以了解 Bootloader 和其相关联的器件复位是如何集成到一个典型的应用（如，风扇控制）内的。PSoC Creator 提供了一个风扇控制器组件，该组件封装了所有必要的硬件模块，包括 PWM、转速计输入捕捉定时器、控制寄存器、状态寄存器和 DMA 通道或中断。有关详细信息，请参考风扇控制器应用页面所介绍的内容。

风扇控制应用位于 Bootloadable 项目中。另外，还可以自定义 Bootloader，从而能在引导加载过程中保持风扇运行。

在互相切换 Bootloader 和 Bootloadable 之间的过程中，如果器件进行复位，风扇仍可以保持运行状态，如表 4 所示。

表 4. 风扇控制器的器件复位时 PSoC I/O 引脚驱动模式

I/O 引脚驱动模式	备注
高阻模拟	在 PWM 引脚上可以加上外部上拉或下拉电阻，使占空比达到 100%。但不必要这样做，因为风扇由于其惯性可以保持转动。
NVL 设置：高阻模拟、上拉或下拉	可以将 PWM 引脚组件的复位值设置为上拉或下拉，从而得到 100%的占空比。但不必要这样做，因为风扇由于其惯性可以保持转动。
PSoC Creator 项目配置	设置 PWM 引脚组件驱动模式和初始状态，从而得到 100%的占空比。 PWM 组件变为有效，但不运行。
Main()开始执行	PWM_Start()被调用时，PWM 会以组件的默认占空比来开始驱动 PWM 引脚。 固件可以读取转速计数据，并主动控制占空比。

文档修订记录

文档标题： AN73854 — PSoC® 3、PSoC 4 和 PSoC 5LP 中 Bootloader 的简介

文档编号： 001-97048

版本	ECN	变更者	提交日期	变更说明
**	4723006	HENG	05/06/2015	本文档版本号为 Rev**，译自英文版 001-73854 Rev*G。
*A	5027479	HENG	12/10/2015	本文档版本号为 Rev*A，译自英文版 001-73854 Rev*I。
*B	5716205	AESATMP9	04/28/2017	更新徽标和版权。

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

赛普拉斯开发者社区

论坛 | WICED IoT 论坛| 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

此处引用的所有其他商标或注册商标归其各自所有者所有。

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

赛普拉斯半导体公司，2011-2017 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。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。