PSoC 4 I2C Bootloader

关于本文档

范围和目的

AN86526 介绍了 PSoC™ 4 中基于 I^2C 的 Bootloader。通过本应用笔记，您可以了解如何使用 PSoC™ Creator 快速轻松地编译一个基于 I^2C 的 Bootloader 项目和 Bootloadable 项目。同时，它还介绍了如何编译一个基于 I^2C 的嵌入式 Bootloader 主机程序。

相关器件系列：PSoC™ 4000, 4100, 4200, 4000S, 4100S, 4100S Plus, 400PS

软件版本：PSoC™ Creator 4.2 SP2 及更高版本

相关应用笔记：**请点击此处。**

目标读者

本文档的目标读者是使用 PSoC™ Creator 为 PSoC™ 4 创建 I2C 引导器的用户。

更多代码示例？我们倾听到了你的声音。

要获取各种应用并且实时更新的示例程序，请访问我们的**示例程序网站**，你也可以在**这里**观看关于视频教程。

目录

关于本文档 1
目录 1
1 简介 3
1.1 术语和定义 3
1.2 使用 Bootloader 4
1.3 Bootloader 功能流程 4
1.4 进入 Bootloader 的方法 6
1.4.1 Bootloadable API 6
1.4.2 自定义 Bootloader 6
2 项目 7
2.1 I^2C Bootloader 7
2.2 Bootloadable 12
2.2.1 Bootloadable_Green 项目 — 示例 1 12
2.2.2 使用一个 PC 主机进行引导加载 15
2.2.3 Bootloadable_Blue 项目 — 示例 2 19
2.3 I2C Bootloader 主机 22
2.3.1 Bootloader 主机程序 22
2.3.2 Bootloader 系统 API 23
2.3.3 创建 I2C Bootloader 主机项目的步骤 23
3 测试项目 27
3.1 配置套件 27
3.2 验证结果 28

目录

4 总结 29
5 相关应用笔记 30
6 相关项目 31
7 PSoC™资源 32
7.1 PSoC™ Creator 32
7.2 代码示例 33
7.3 PSoC™ Creator 帮助 35
7.4 技术支持 35
8 附录 A：存储器 36
8.1 闪存存储器的详细信息 36
8.2 PSoC™中存储器的使用情况 37
8.3 内存中的元数据布局 38
8.3.1 闪存保护 39
8.3.2 闪存保护的示例 39
9 附录 B：项目文件 41
9.1 Bootloadable 输出文件 41
9.1.1 *.cyacd 文件格式 41
10 附录 C：主机/目标通信 42
10.1 通信流程 42
10.2 协议数据包格式 43
10.3 Bootloader 主机的 I2C 数据操作信息 44
10.3.1 指令和状态/错误代码 45
11 附录 D：主机内核 API 47
11.1 cybtldr_api2.c / .h 47
11.2 cybtldr_parse.c / .h 47
11.3 cybtldr_api.c / .h 47
11.4 cybtldr_command.c / .h 47
12 附录 E：其它主题 48
12.1 Bootloader 与 HSSP 48
12.2 在引导加载过程中，如果发生断电，将导致什么情况？ 48
12.3 在 Bootloader 和 Bootloadable 项目之间进行跳转，为什么需要复位操作？ 48
12.4 将应用项目类型从普通 (Normal) 转换为 Bootloadable 48
12.5 调试 Bootloadable 项目 48
12.6 多应用 Bootloader 48
12.6.1 Bootloader 的存储器要求 50
12.6.2 PSoC™中 PSoC™套件与 MiniProg3 的对比 51
13 附录 F：套件选择 52
文档修订记录 53

1 简介

Bootloader (引导加载程序）是 MCU 系统设计中的常用部分。通过 Bootloader，可以在现场更新产品的固件。在工厂里，初步将固件编程到某个产品内时，通常是通过 MCU 的 JTAG 或 Arm®串行线调试器 (SWD）接口实现的。但这些接口一般不能在现场访问。

这样就需要实现引导加载过程。引导加载过程允许通过一个标准的通信接口 (如 USB、I²C、UART 或 SPI）对您的系统固件进行更新。Bootloader 通过与主机通信获取新的应用代码或数据，然后将这些代码或数据写入到器件的闪存中。

本应用笔记包含以下主题：

- 如何使用 PSoC™ Creator 创建一个 I²C Bootloader
- Bootloader 主机的相关主题包括：
  - 如何使用 Bootloader 主机工具
  - Bootloader 主机系统的基本编译模块及功能
  - 如何使用 PSoC™创建一个嵌入式 I²C Bootloader 主机

本应用笔记假定您已经熟悉了解了 PSoC™ 4 器件和 PSoC™ Creator 集成开发环境 (IDE)。如果您尚未了解 PSoC™ 4，请参考 **AN79953 — PSoC™ 4 入门手册**。如果您尚未了解 PSoC™ Creator，请参考 **PSoC™ Creator 主页**。

本应用笔记同时假设您已经熟悉了有关 Bootloader 的概念。如果您还不太熟悉这些概念，请参考 **AN73854 — PSoC™ 3、PSoC™ 4 和 PSoC™ 5LP 中 Bootloader 的简介**。要获取有关引导加载过程的完整应用笔记列表，请单击此处。

最后，本应用笔记假设您已经熟悉了 I²C 协议以及 PSoC™ Creator I²C (SCB 模式）组件。如果您对该组件还不太熟悉，请参考 **PSoC™ 4 串行通信模块 (SCB）组件数据手册**中的内容。您也可以通过右键点击 PSoC™ Creator 中的某个 I²C (SCB 模式）组件，获得所需要的数据手册。更多有关 PSoC™ Creator 的信息，请参考以下各节。

1.1 术语和定义

Figure 1 介绍了 Bootloader 系统中的各主要部分。通过该图可以知道，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 — 正常操作和更新闪存存储器操作。用于更新闪存的嵌入式固件部分被称为 **Bootloader**。

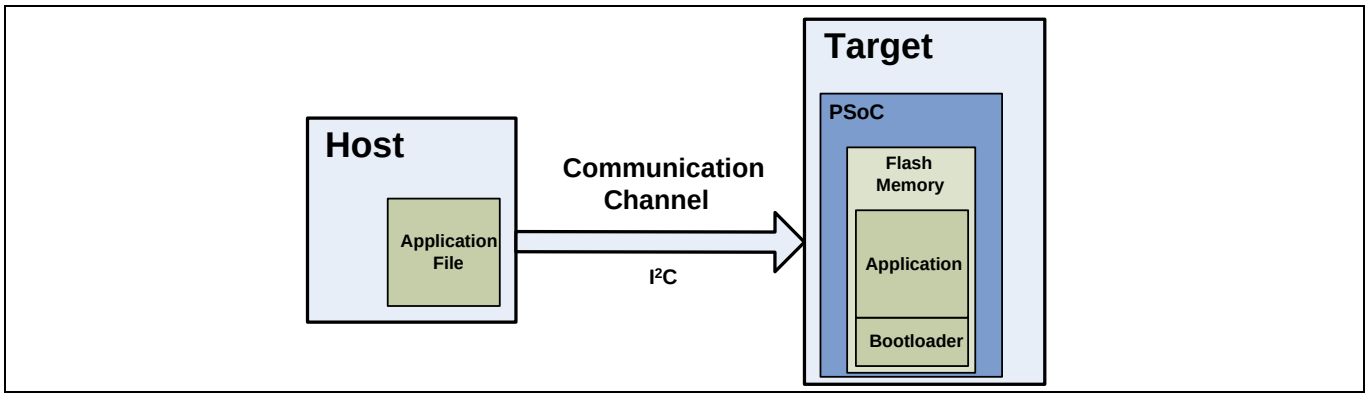


Figure 1 引导加载系统框图

提供用于更新闪存数据的系统被称为**主机 (Host)**，被更新的系统为**目标机 (Target)**。主机可以为外部 PC (PC 主机) 或另一个 MCU (嵌入式主机)。

简介

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，或者简称为引导加载**。放置在闪存中的固件被称为**应用程序或可引导加载程序 (Bootloadable)**。

引导加载的另一个常见术语是系统内编程 (ISP)。英飞凌有一个与 ISP 名称相似而功能相异的产品，即“系统内串行编程器 (ISSP)”，它的操作被称为“主机源串行编程 (HSSP)”。更多信息，请参考 **AN84858 – 使用外部微控制器 (HSSP) 编程 PSoC™ 4** 。

1.2 使用 Bootloader

Bootloader 和实际应用程序通常共用一个 Bootloader 通信端口。要想使用 Bootloader，首先对目标进行操作，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) 指令。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指令，然后把这些指令发送到 Bootloader 中。发送完整个文件后，Bootloader 将控制权交给新的应用程序。

1.3 Bootloader 功能流程

一般来说，当器件复位时，Bootloader 是第一个被执行的功能。然后它将执行下面的操作：

- 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/闪存更新的操作
- 将控制权交给应用程序

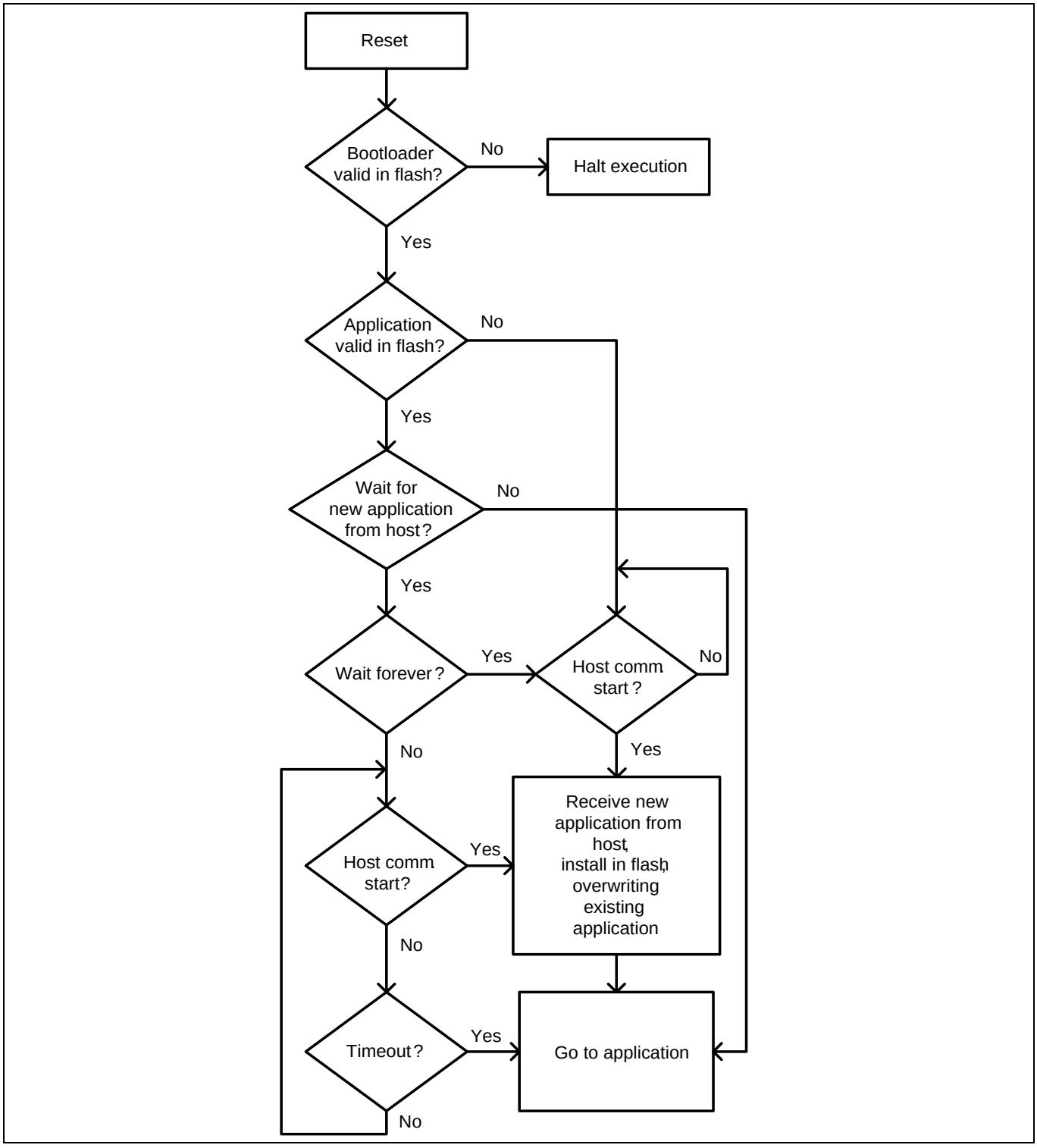


Figure 2　引导加载过程流程图

1.4 进入 Bootloader 的方法

如前面所述，Bootloader 是复位时要运行的第一个功能。如 **Figure 2** 所示，将控制权交给应用程序前，Bootloader 代码会在较短的时间内等待主机。这样可能使主机错失启动引导加载操作的机会。但还存在另一种启动引导加载操作的方法，即将控制权从应用程序或 Bootloadable 交给 Bootloader。

1.4.1 Bootloadable API

PSoC™ Creator 中的 Bootloadable 组件有一个 API 函数，`Bootloadable_Load()`，用于启动 Bootloader。这样主机可以随时启动一个引导加载操作。

该方法的缺点是您必须使用应用程序代码来进行应用程序升级。如果应用有缺陷，会阻止将控制权交给引导加载程序，这时会发生什么？

1.4.2 自定义 Bootloader

相反，使 Bootloader 无限期等待主机可能会更好。为实现该目的，在调用 `Bootloader_Start()`和运行其正常子程序前，可以通过自定义 Bootloader 项目检查某些用户输入。

例如，调用 `Bootloader_Start()`前，Bootloader 可能会监控 UART，并始终等待某个用户指令。更多有关信息，请参考 **AN73854 – PSoC™ 3、PSoC™ 4 和 PSoC™ 5LP 中 Bootloader 的简介**。

2 项目

本节列出的各个步骤可创建以下 PSoC™ Creator 项目：

- I²C Bootloader
- Bootloadable
- 嵌入式 Bootloader 主机

这些项目与英飞凌开发套件一起使用。开发套件是根据目标器件选择的；请参考**附录 F：套件选择**了解详细信息。您可能要根据开发套件更改引脚连接。请查看特定的套件文档，了解有关连接的信息。然而，这些项目容易适应其他自定义电路板。

2.1 I²C Bootloader

本节介绍了如何创建并编译一个基于 I²C 的 Bootloader 项目。该项目的特点之一是进行引导加载时，套件中的红色 LED 会闪烁发光。下面示例描述了使用 PSoC™ 4200 器件的 I²C Bootloader 项目，但是该流程与所有其它 PSoC™器件完全相同。

1. 创建一个新的 PSoC™ Creator 项目，并将其命名为“I2C_Bootloader_Red”。选择目标 PSoC™器件并为该项目创建一个新的工作区。

注意：对于 PSoC™ Creator 3.1 以及更低的版本，当创建项目时应指定“应用类型”。要想指定应用类型，请点击 **Advanced** 选项卡旁边的‘+’按键来扩展配置选项。将 Bootloader 选择为应用类型。

2. 分别将一个 I²C (SCB 模式) 和一个 Bootloader 组件添加到顶层设计原理图内。为了使 LED 闪烁，请添加一个 PWM (TCPWM 模式)、一个时钟和一个数字输出引脚组件。根据 **Table 1** 中显示的内容重命名各个组件。

Table 1 I2C_Bootloader_Red 项目组件名称

组件	名称
Bootloader_1	Bootloader
I2C_1	I2C
PWM_1	PWM
Clock_1	Clock
Pin_1	Pin_LED

3. 要想配置 Bootloader，请双击组件。选择 **I2C** 作为**通信组件**，如 **Figure 3** 所示。保留其他参数的默认设置。更多有关这些配置参数的信息，请参考 **Bootloader 组件数据手册**中的内容。

*Note: 将 **Wait for command time** (等待指令时间) 参数设置为 2 秒。有关 Bootloader 等待时间的详细信息，请参考 Bootloader 组件数据手册中“Wait for Command Time” (等待指令的时间) 的内容。*

项目

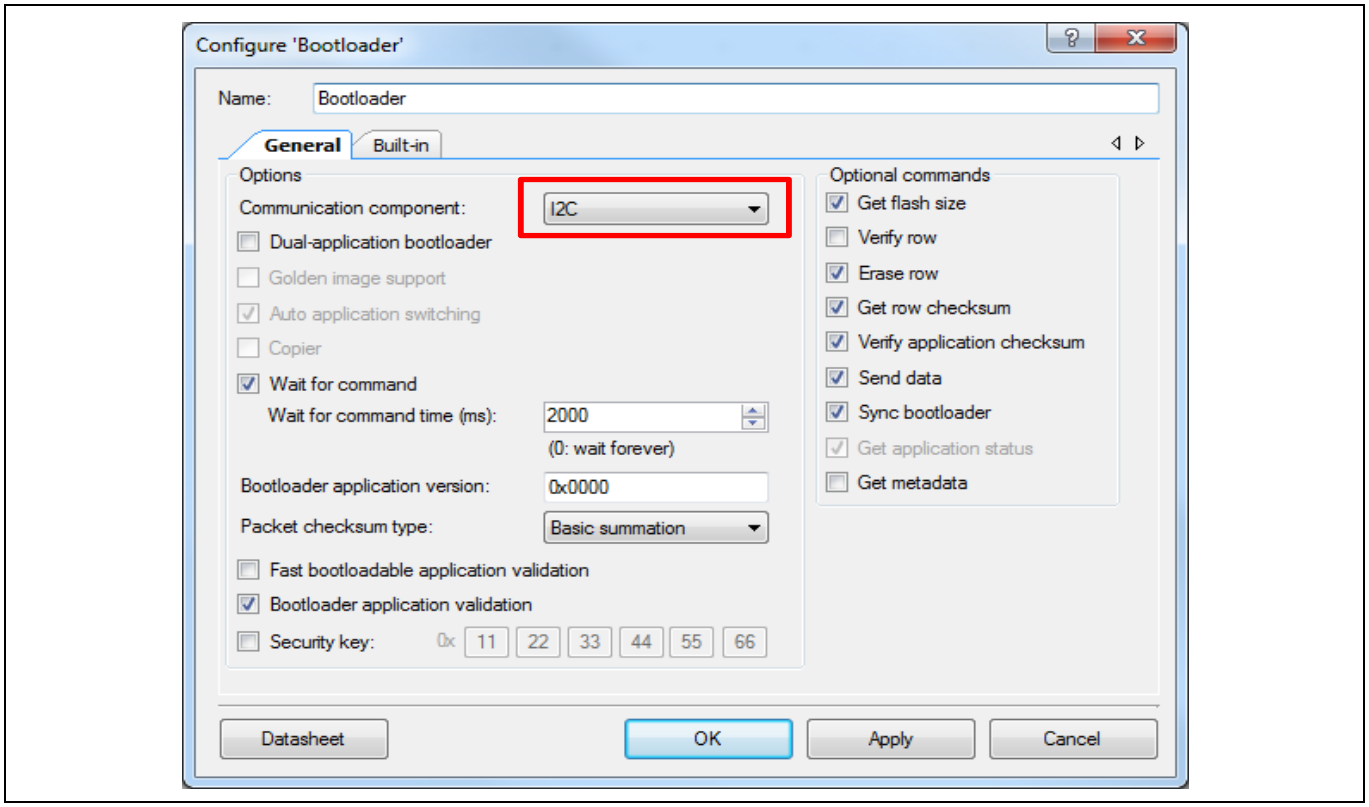


Figure 3 Bootloader 配置

4. 想要配置 I²C (SCB 模式) 组件，请双击它。默认情况下，该模式被设置为“Slave”，数据速率为 100 kbps，从地址 (slave address) 为 8。如果您想使用其他地址，请将其输入到 **Slave address** 框中。保留其他参数的默认设置。

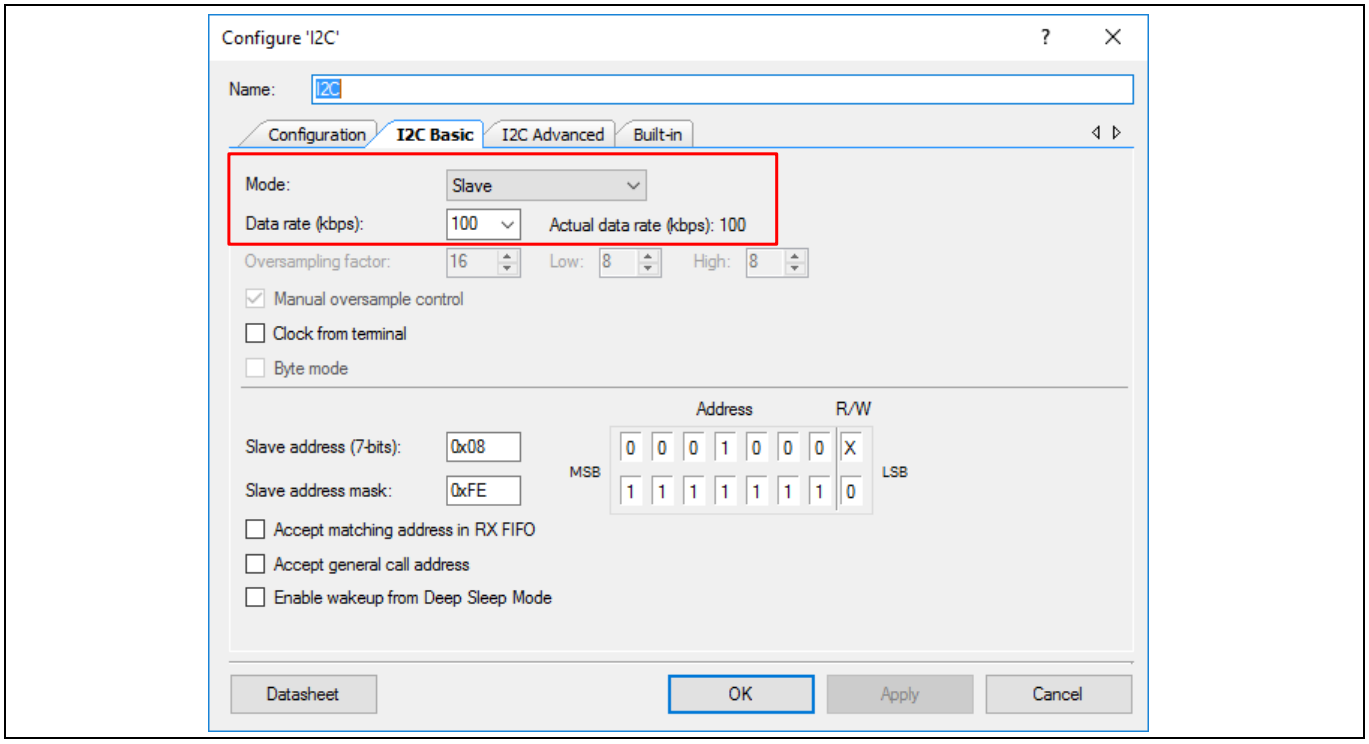


Figure 4 基本 I2C (SCB 模式) 配置

项目

该 I²C (SCB 模式) 组件包含各种引脚组件，这些引脚组件被自动配置，从而能够与 I²C 总线 (开漏，驱动低电平) 一起使用。您必须添加外部上拉电阻，如第·10 页上的 **Figure 8** 所示，并将它们分配给各个物理引脚 (即第 10 页上的步骤 8)。

5. 要配置 PWM (TCPWM 模式) 组件，请双击它。将 **Period** (周期) 项设置为 254，并将 **Compare** (比较) 项设置为 127。保留其他参数的默认设置。

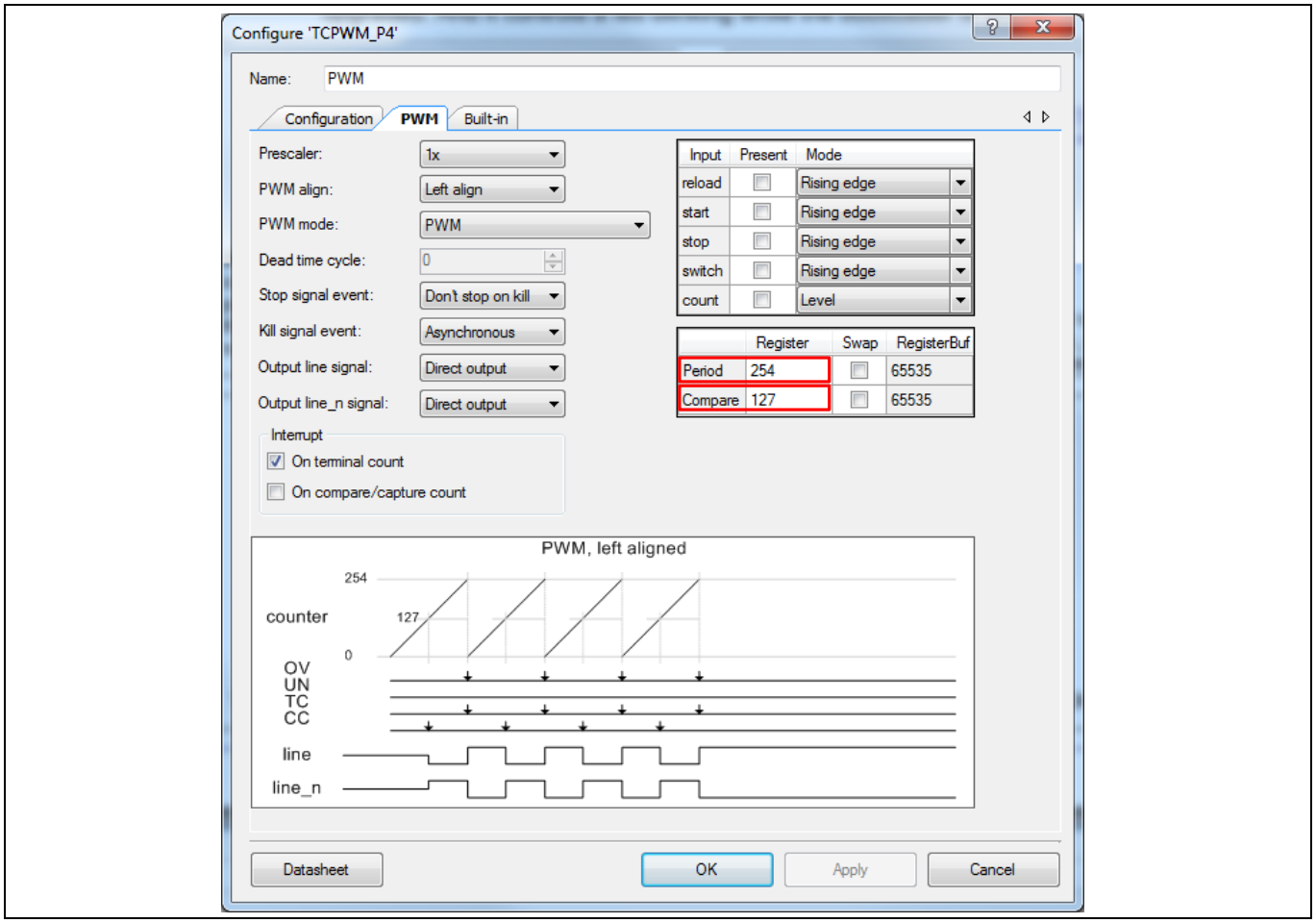


Figure 5 PWM (TCPWM 模式) 的基本配置

6. 对于引脚组件，保留它们的默认设置情况。使用导线将引脚连接到 PWM 组件的“line” (线) 终端。

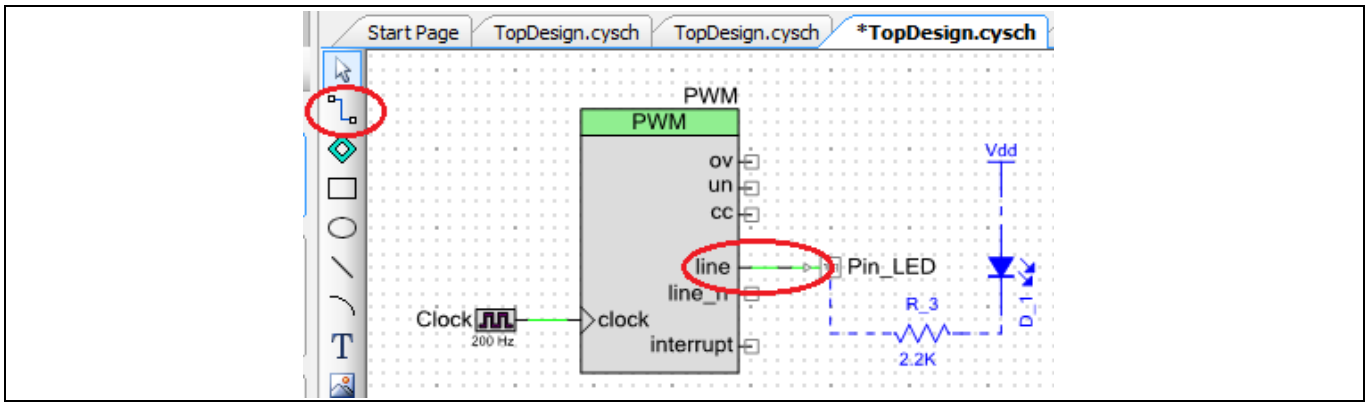


Figure 6 引脚连接

项目

7. 要想配置 Clock 组件，请双击它。将 **Frequency** 项设置为 200 Hz。保留其他参数的默认设置。

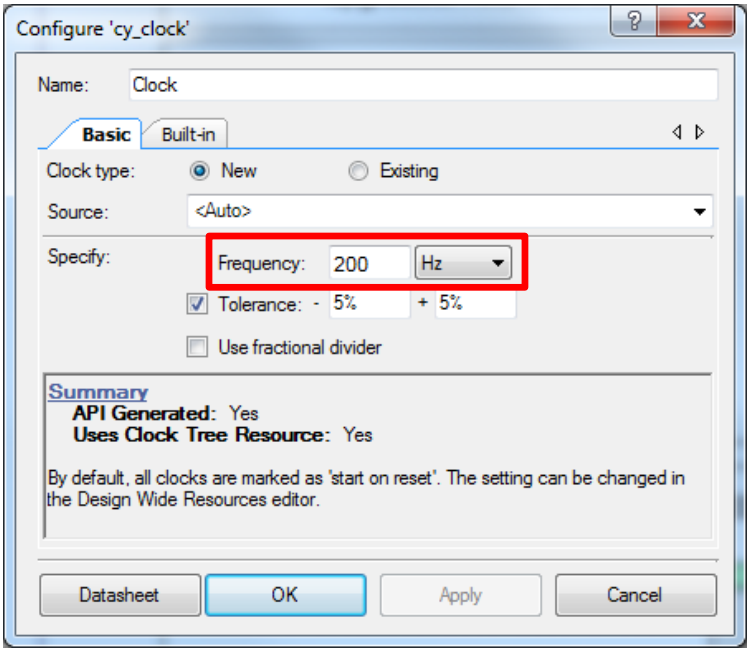


Figure 7 时钟配置

添加了电阻和 LED 等注释组件后，该项目的顶层设计与 **Figure 8** 所显示的情况相似。

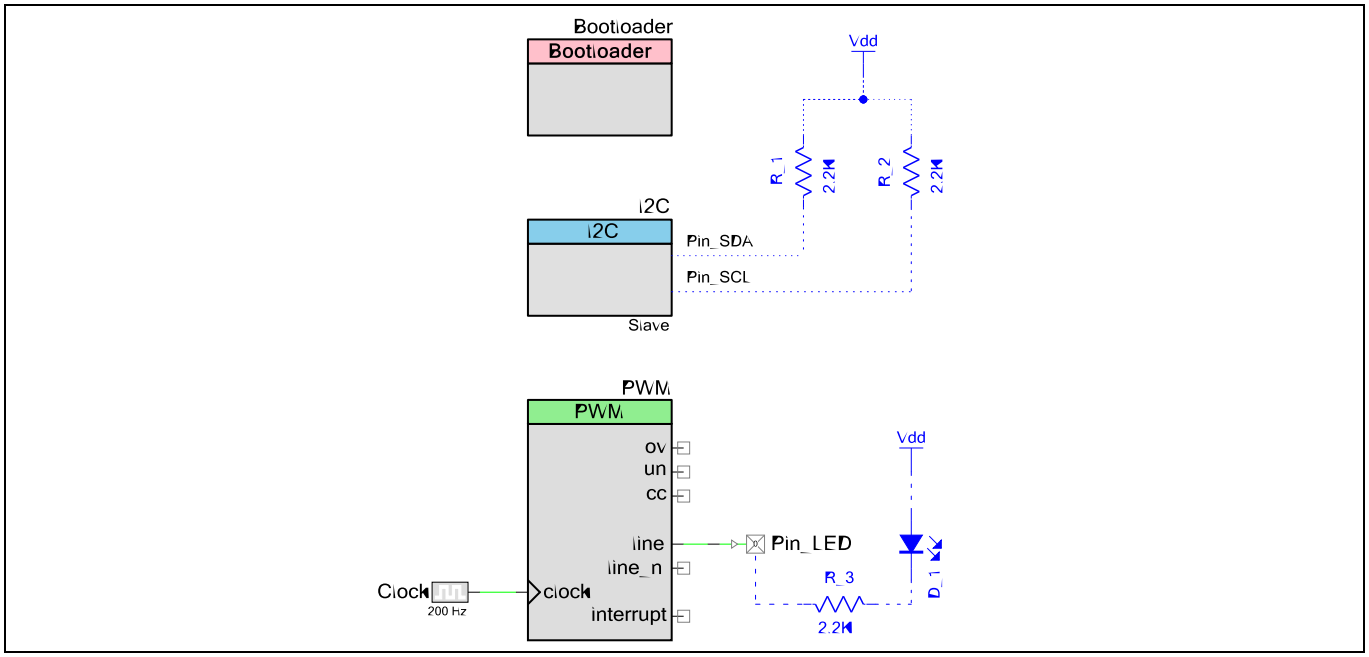


Figure 8 I2C_Bootloader_Red 项目的顶层设计

项目

8. 将 I²C 引脚组件分配给各个物理引脚。在 **Workspace Explorer** (工作区浏览器) 窗口中，双击 *I2C_Bootloader_Red.cydwr* 文件，并点击 **Pins** (引脚) 选项卡。引脚分配由器件决定。请参考器件数据手册了解更多信息。例如，对于 **CY8CKIT-042** 上的 **PSoC™ 4200** 器件，请按 **Figure 9** 分配各引脚；对于 **CY8CKIT-040** 上的 **PSoC™ 4000** 器件，则按 **Figure 10** 分配各引脚。

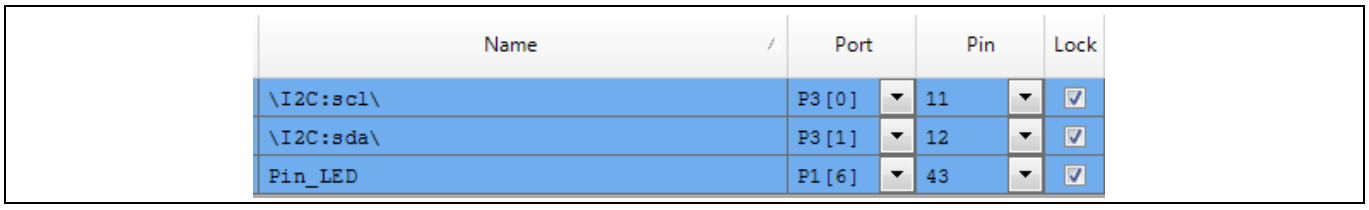


Figure 9 CY8CKIT-042 的 I2C 引脚分配

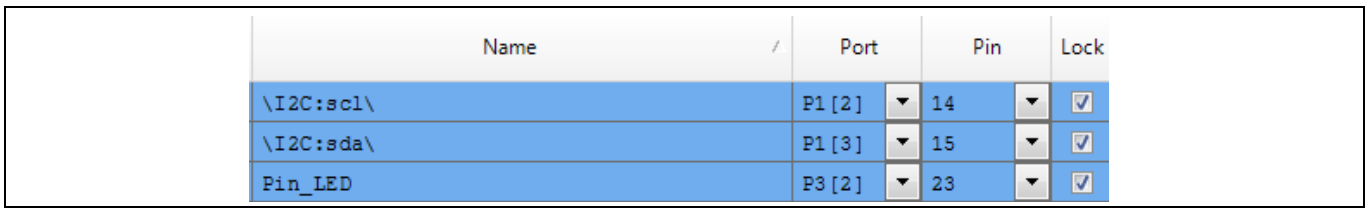


Figure 10 CY8CKIT-040 的 I2C 引脚分配

9. 将 `Bootloader_Start()`函数添加到 `main()`函数内。该 API 函数将执行整个引导加载过程。它不会返回，但会在器件软件复位时结束。因此，始终不会执行该函数后的代码。添加两个 API 函数，用于初始化 `main()`中的 PWM，如 **Code Listing 1** 所示。更多有关该组件 API 的信息，请参考 **TCPWM 组件数据手册**。

Code Listing 1 Bootloader 中的 PWM 初始化

```
Intmain()
{
  /* Initialize PWM */
  PWM_Start();
  PWM_TriggerCommand(PWM_MASK,
                     PWM_CMD_START);

Bootloader_Start();

for(;;)
  {
    /* Place your code here. */
  }
}
```

10. 编译该项目并将其编程到根据目标器件选择的特定套件中。

项目

2.2 Bootloadable

现在我们要创建两个 Bootloadable 项目。第一个项目会使 PSoC™开发套件上的绿色 LED 闪烁。第二个项目将演示如何从一个 Bootloadable 项目进入一个 Bootloader 项目，并使该套件上的蓝色 LED 闪烁。

*Note: **CY8CKIT-042** 和 **CY8CKIT-040** 上都有 RGB LED，如 **Figure 31** 和 **Figure 32** 所示。在这里，绿色 LED 和蓝色 LED 用于指示哪个项目正在运行。*

2.2.1 Bootloadable_Green 项目 – 示例 1

本节介绍了创建第一个 Bootloadable 项目的各个步骤。

1. 创建一个应用类型为 Bootloadable 的新 PSoC™ Creator 项目。将该项目命名为 **Bootloadable_Green**。该项目和 **I2C_Bootloader**_Red 项目必须使用相同的器件。
2. 对于该项目，您需要使用 Bootloadable 和数字输出引脚组件。将这两个组件添加到您的顶层设计原理图中。根据 **Table 2** 中显示的内容命名这些组件。

Table 2 Bootloadable_Green 项目中各组件的名称

组件	名称
Bootloadable_1	Bootloadable
Pin_1	Pin_LED

3. 要想配置 Bootloader 组件，请双击它。

Bootloadable 项目需要始终链接到 Bootloader 项目的*.hex* 文件。要想链接该项目，请打开 **Dependencies** 选项卡，然后将 Bootloadable 链接到 *I2C_Bootloader_Red.hex* 文件，如 **Figure 11** 所示。更多有关 Bootloadable 组件配置的信息，请参考 **Bootloadable 组件数据手册中的内容**。

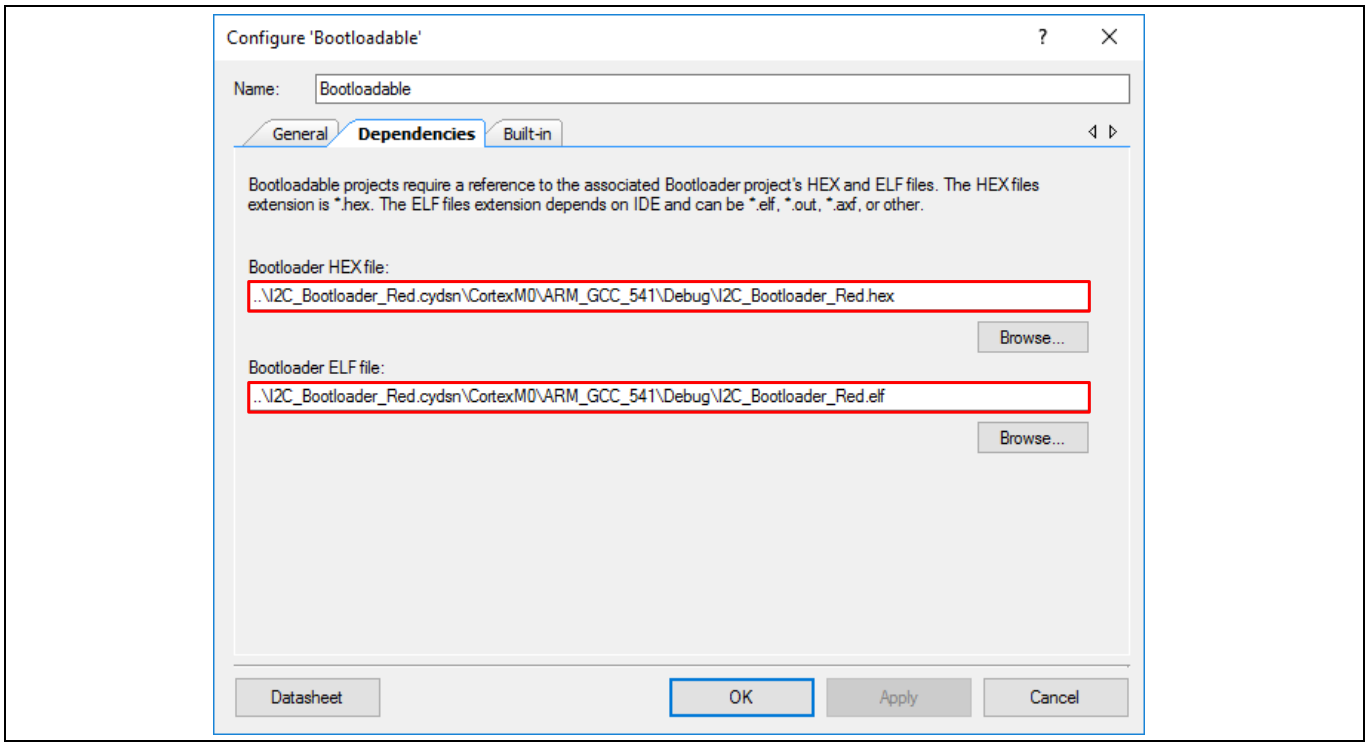


Figure 11 Bootloadable 组件配置

项目

您可以在 Bootloader 项目的 Debug 或 Release 文件夹内查找 *I2C_Bootloader_Red.hex* 文件：

1. 对于 PSoC™ 4000, 4100, 4200 器件

..\I2C_Bootloader_Red.cydsn\CortexM0\ARM_GCC_541\Debug\I2C_Bootloader_Red.hex

2. 对于 PSoC™ 4000S, 4100S, 4100S Plus 和 4100PS

..\I2C_Bootloader_Red.cydsn\CortexM0p\ARM_GCC_541\Debug\I2C_Bootloader_Red.hex

3. 要想配置引脚组件，请双击它。取消选择 **HW Connection** 项，如 **Figure 12** 所示。将驱动模式设置为 **Strong Drive**，如 **Figure 13** 所示。

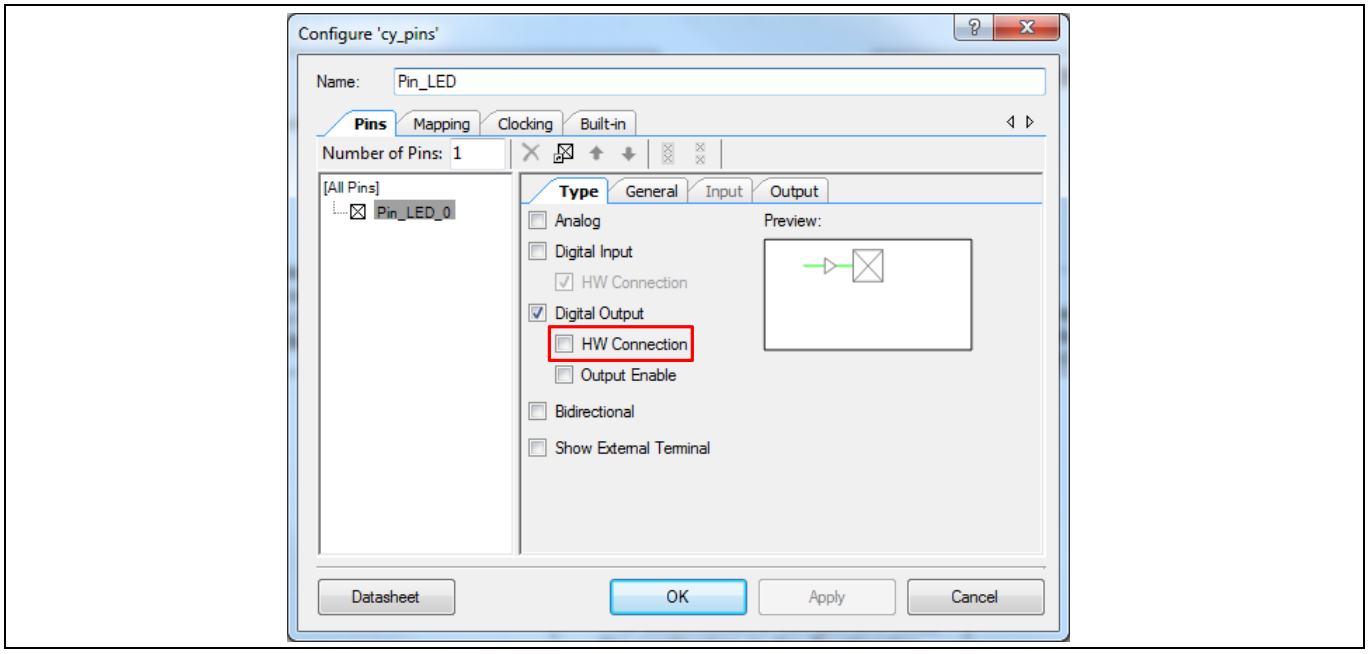


Figure 12　引脚组件配置 — Type 选项卡

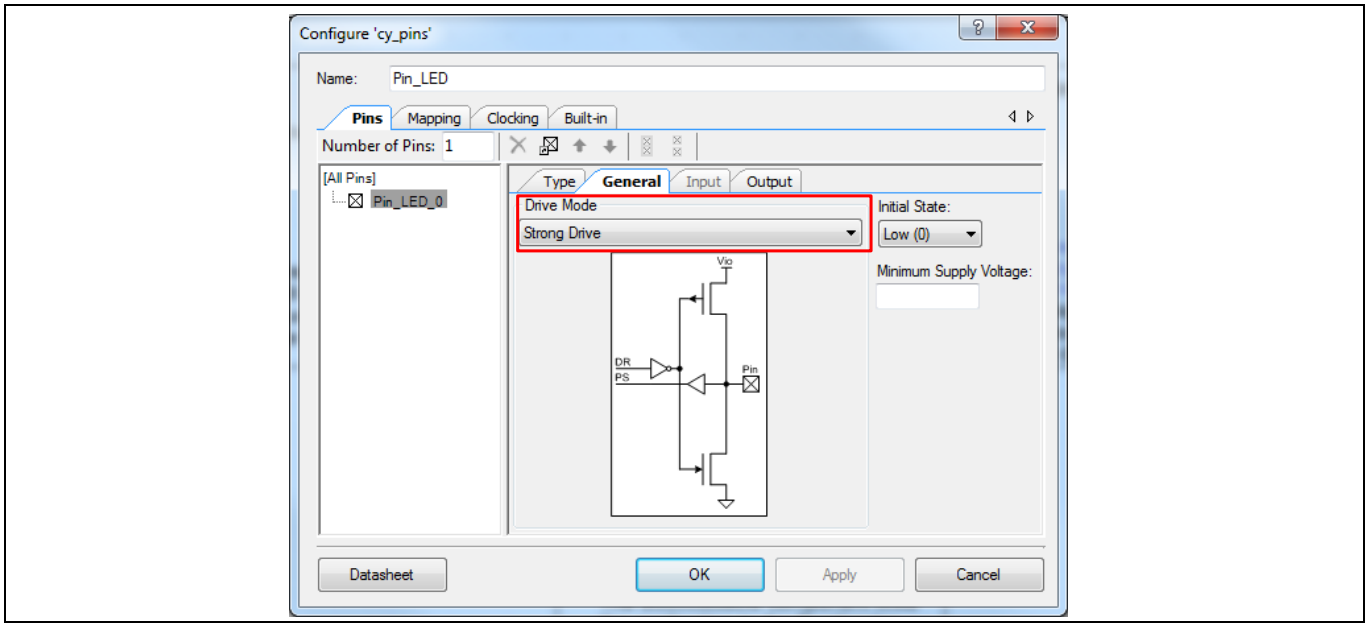


Figure 13　引脚组件配置 — General 选项卡

项目

添加好 LED 的注释组件后，便完成了顶层设计。它应该与 **Figure 14** 相似。

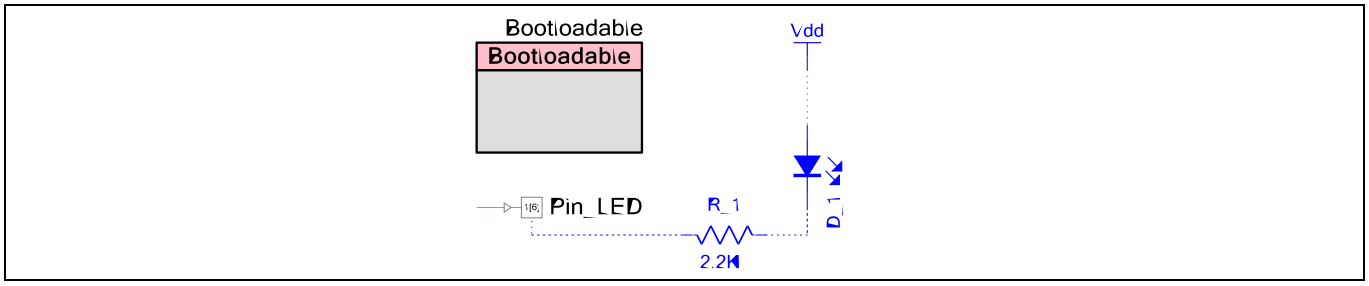


Figure 14 Bootloadable_Green 项目的 顶层设计

4. 将下面的代码添加到 *main.c* 中，从而使 LED 闪烁：

```
Voidmain()
{
for(;;)
  {
      /* Toggle the LED */
Pin_LED_Write(~Pin_LED_Read());

      /* Delay 1 second */
CyDelay(1000u);
  }
}
```

5. 将引脚组件分配给物理引脚。在 **Workspace Explorer** 窗口中，双击 *Bootloadable_Green.cydwr* 文件并分配引脚。更多有关引脚分配的信息，请参考套件用户指南。例如，**Figure 15** 显示的是 **CY8CKIT-042** 套件电路板的引脚分配，而 **Figure 16** 则显示的是 **CY8CKIT-040** 套件电路板的引脚分配。



Figure 15 CY8CKIT-042 Bootloadable_Green 项目的引脚分配



Figure 16 CY8CKIT-040 Bootloadable_Green 项目的引脚分配

项目

6. 编译项目。编译该 Bootloadable 项目时，PSoC™ Creator 将生成一个.*cyacd* 文件。该文件将被引导加载到目标内。更多有关该文件及其内容的信息，请参考**附录 B**。

现在，我们使用 PSoC™ Creator 将该项目引导加载到 PSoC™中。

2.2.2 使用一个 PC 主机进行引导加载

通过 PSoC™ Creator 附带的 Bootloader 主机可执行文件，可以从 PC 主机上引导加载一个应用程序。PSoC™ 4 套件使用板上 PSoC™ 5LP 来实现 USB-I²C 桥接器。该桥接器用于将 PC 连接至 Bootloader，如 **Figure 17** 中的 **CY8CKIT-042** 套件和 **Figure 18** 中的 **CY8CKIT-040** 套件所示。

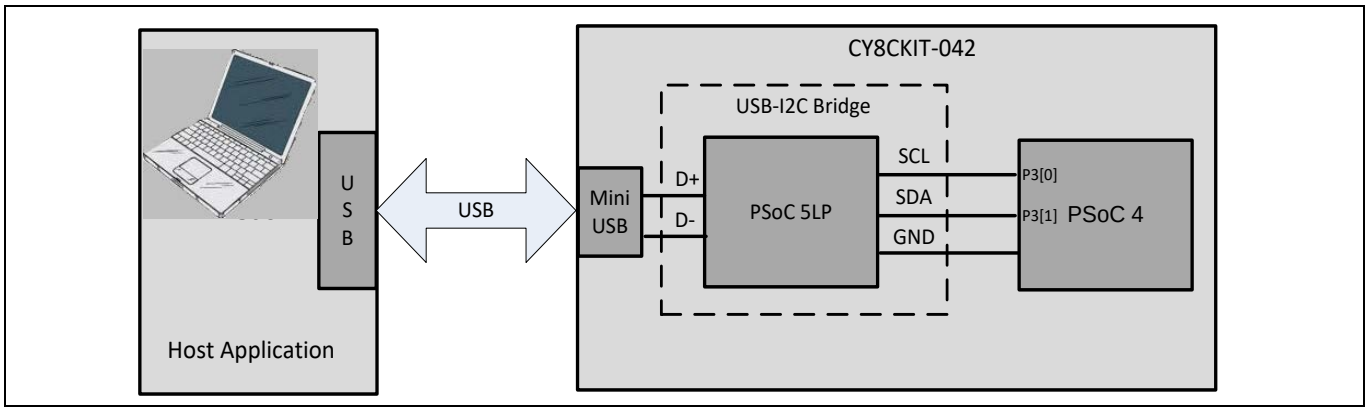


Figure 17 使用一个 PC 主机进行引导加载 (CY8CKIT-042)

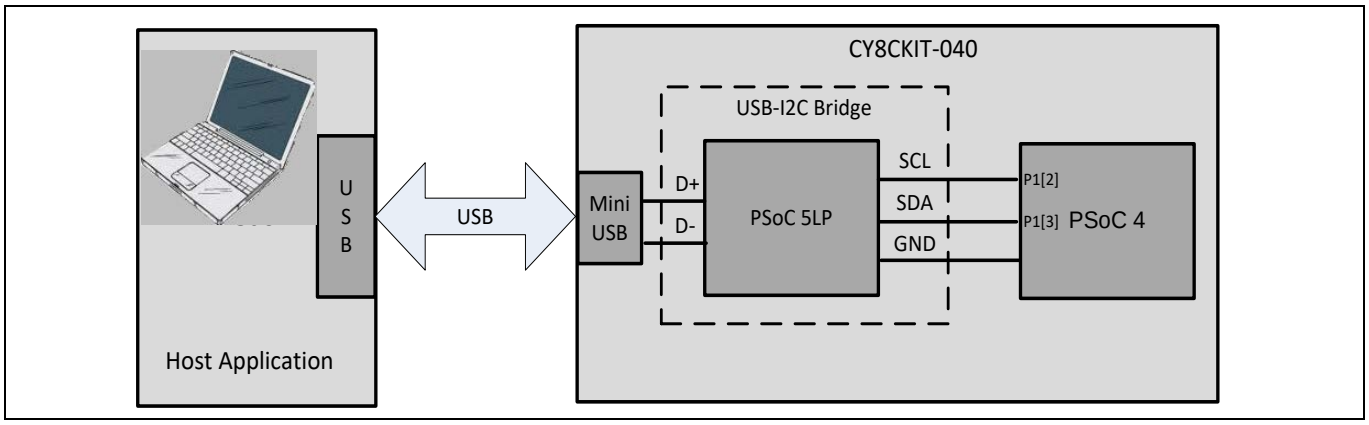


Figure 18 使用一个 PC 主机进行引导加载 (CY8CKIT-040)

Note: 在所有 PSoC™ 4 套件中，各个引脚与板上 PSoC™ 5LP SCL 相连，另外 SDA 引脚上带有内部上拉电阻。因此，不需要使用外部上拉电阻。更多有关 I²C 引脚连接的信息，请参考套件用户指南。

*Note: **另外，还可以将 CY8CKIT-002 MiniProg3** 作为 USB-I²C 桥接器使用，从而进行引导加载。更多有关信息，请参考 **MiniProg3 用户指南**和基础知识文章**通过 I2C** 引导加载时的 MiniProg3 连接。*

下面各步骤介绍的是 PSoC™ 4200 器件使用 Bootloader 主机程序引导加载应用程序的过程。该程序与 PSoC™系列中其它器件的程序完全相同。如前面所介绍，在启动引导加载操作前，您必须将 Bootloader 项目编程到 PSoC™器件内。

1. 首先，使用 USB 线缆 (同时具有供电作用) 将 **CY8CKIT-042** 连接到 PC 上，如 **Figure 19** 所示。

项目

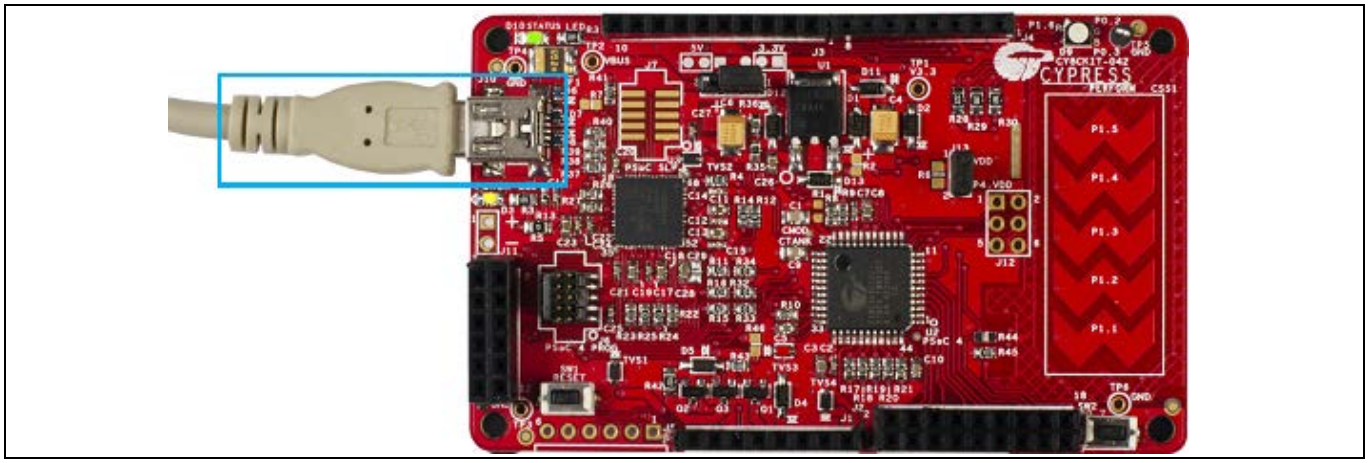


Figure 19　将 USB 线缆连接到 CY8CKIT-042 的 J10 上

2. 在 PSoC™ Creator 中，依次选择 **Tools** >**Bootloader Host**，打开 Bootloader 主机工具。
3. 请确保 Bootloader 主机应用程序的 I^2C 配置 (如 **Figure 20** 所示) 与 Bootloader 项目中的 I^2C (SCB 模式) 组件配置 (**Figure 4**) 相同。

Note:　通过 USB 线缆连接好套件后，如果 Bootloader 主机 GUI 内没有 KitProg 信息，请点击 ***Filters****，以确保使能了* ***Show I2C Devices*** *项。*

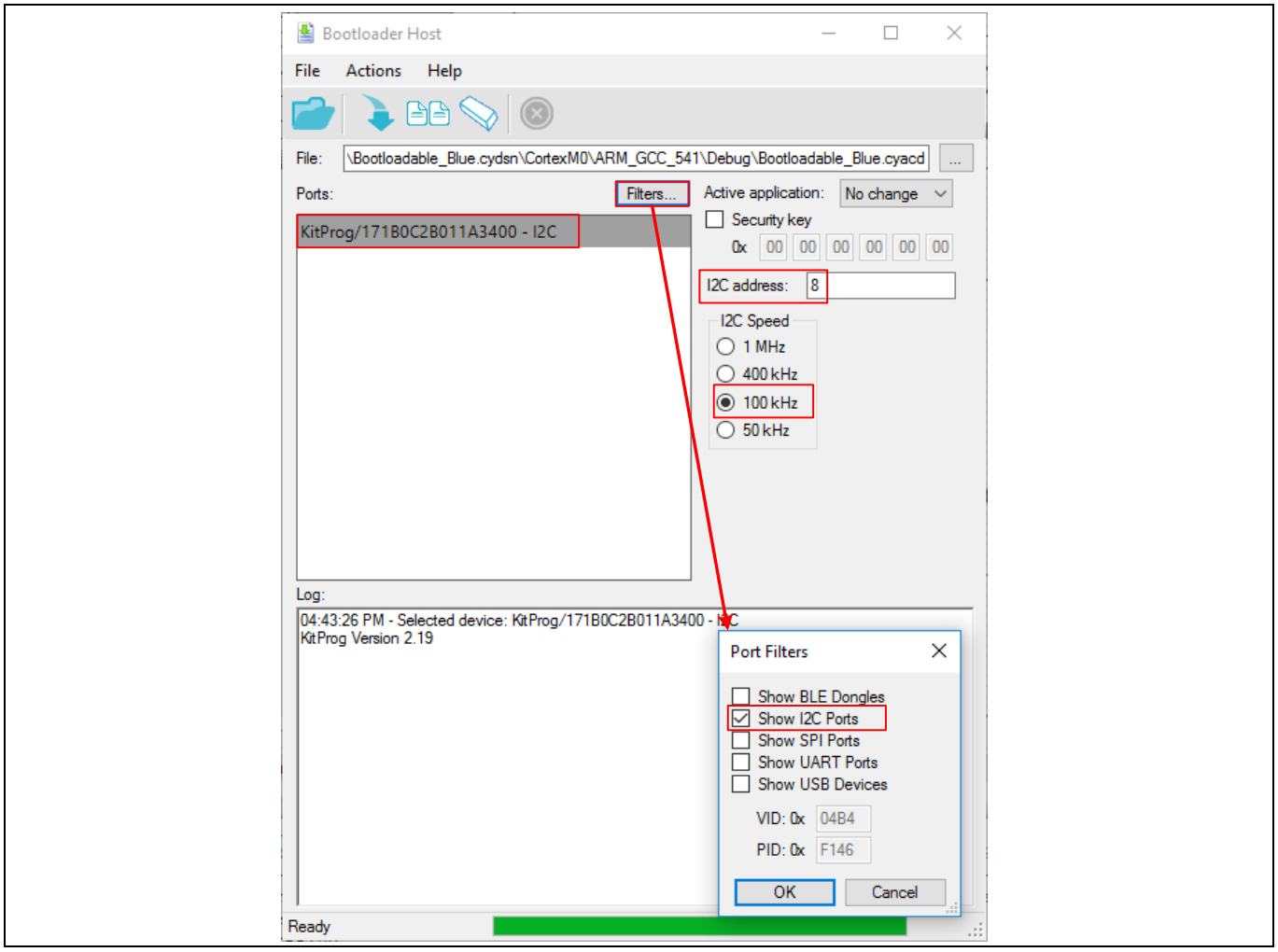


Figure 20 Bootloader 主机应用程序

4. 按下 **File** 按键，然后选择 Bootloadable 项目中 Debug 或 Release 文件夹内的 Bootloadable 文件 *Bootloadable_Green.cyacd*：
   - 对于 PSoC™ 4000, 4100, 4200 器件

     ..\Bootloadable_Green.cydsn\CortexM0\ARM_GCC_541\ Debug\Bootloadable_Green.cyacd
   - 对于 PSoC™ 4000S, 4100S, 4100S Plus 和 4100PS

     ..\Bootloadable_Green.cydsn\CortexM0p\ARM_GCC_541\ Debug\Bootloadable_Green.cyacd

项目

5. 要想引导加载该器件，请点击 **Program** 按键。这时，您的屏幕如 **Figure 21** 所示。

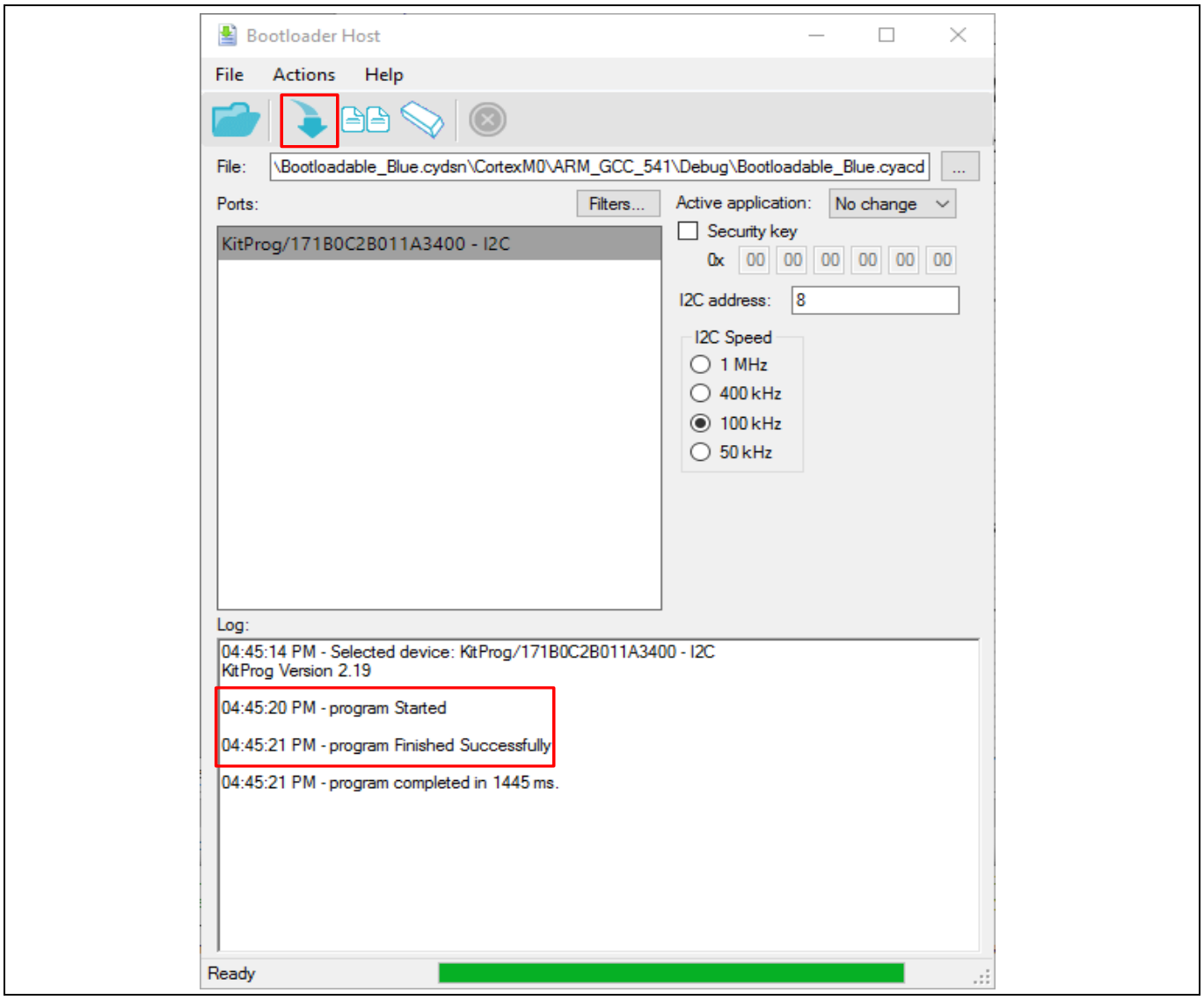


Figure 21　下载 Bootloadable 项目

6. 成功引导加载了 Bootloadable 项目后，将发生软件复位。这时，器件将开始执行新的应用程序。应用程序运行时，套件 RGB LED 闪烁绿色。

要想引导加载另一个应用程序，请通过复位器件 (先按下再释放 PSoC™开发套件的复位按键) 来使能 Bootloader。然后，在 (第 8 页上的 **Figure 3** 中所设置的) **等待指令时间**内按下 Bootloader 主机的 Program 按键。有关 Bootloader 等待时间的详细信息，请参考 **Bootloader 和 Bootloadable 组件数据手册**中“Wait for Command Time” (等待指令的时间) 的介绍。

项目

2.2.3 Bootloadable_Blue 项目 — 示例 2

当 Bootloadable 项目正在运行时，如果您想引导加载一个新项目 (应用程序升级)，那么，Bootloadable 项目可通过调用 API 函数 `Bootloadable_Load()`来启动 Bootloader。

在该示例中，当按下按键时，会调用 `Bootloadable_Load()`函数。本节介绍了创建该 Bootloadable 项目的步骤。

1. 创建一个应用类型为“Bootloadable”的新 PSoC™ Creator 项目 (与**示例 1** 相似)。将该项目命名为“Bootloadable_Blue”。该项目的 PSoC™器件必须与 I2C_Bootloader_Red 项目的器件相同。
2. 对于该项目，需要使用一个 Bootloadable 组件、两个引脚组件以及一个中断组件。将这些组件添加到您的顶层设计原理图内，并根据 **Table 3** 进行命名。

Table 3 Bootloadable 组件名称

组件	名称
Bootloadable_1	Bootloadable
Pin_1 (数字输入引脚)	Pin_StartBootloader
Pin_2 (数字输出引脚)	Pin_LED
isr_1	isr_EnterBootloader

3. 根据 **Figure 11**，将 Bootloadable 组件连接到 Bootloader 项目。
4. 数字输入引脚 (即 Pin_StartBootloader) 用于将应用程序切换回 Bootloader。当按下连接到该引脚的按键时，通过调用 API 函数 `Bootloadable_Load()`，应用程序可以进入 Bootloader。Bootloader 无限期等待，直到主机开始进行引导加载操作为止。

   按下开发套件上的按键时，该引脚被短路接地，因此，请将它的驱动模式配置为 **Resistive Pull Up** (上拉电阻)，如 **Figure 22** 所示。此外，对它进行设置，使之在**下降沿**到来时生成一个中断，如 **Figure 23** 所示。按下按键 (而不是释放它) 时，将生成中断。

   最后，点击 Type 选项卡，并禁用 HW Connection 项。

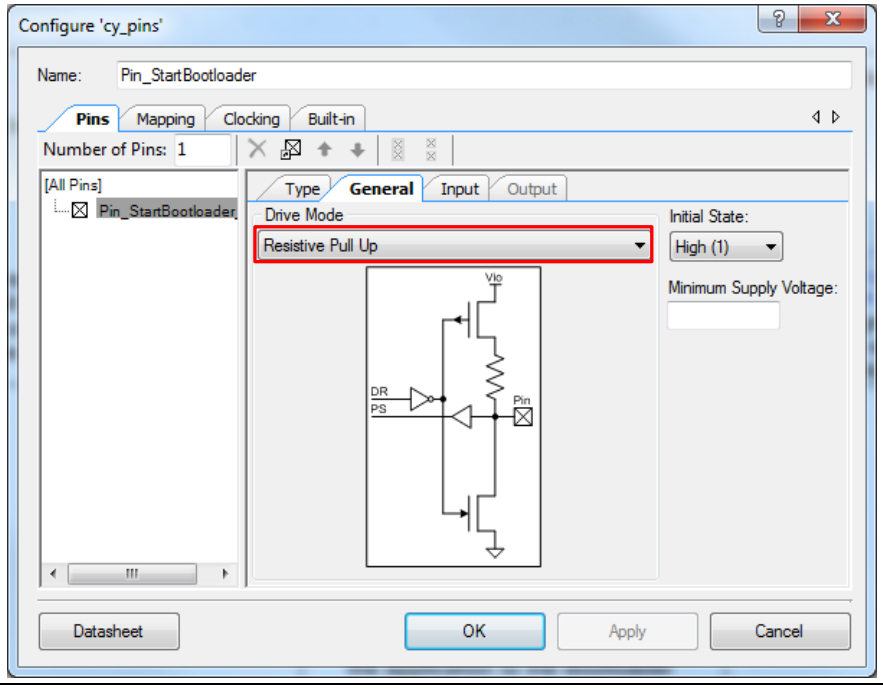


Figure 22 数字输入引脚配置

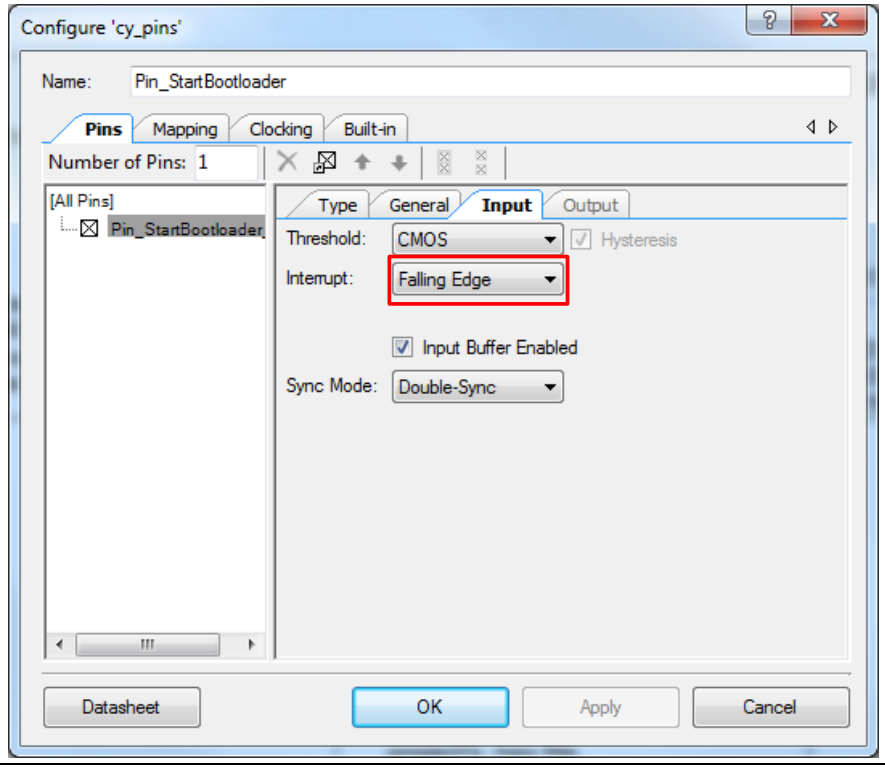


Figure 23　数字输入引脚配置

5. 数字输出引脚 Pin_LED 用于控制 LED。它的设置情况与**示例 1** 中 Pin_LED 的相同；请参考 **Figure 12** 和 **Figure 13** 中的内容。
6. 将 ISR 组件 isr_EnterBootloader 连接到 Pin_StartBootloader 的中断终端 (irq) 上。添加了按键和 LED 的注释组件后，将完成顶层设计；它应该与 **Figure 24** 所示的内容相似。

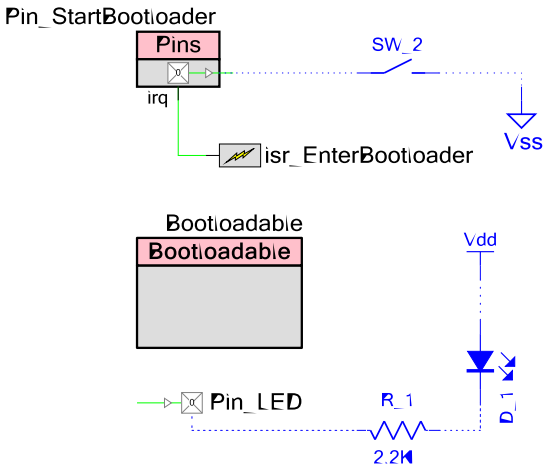
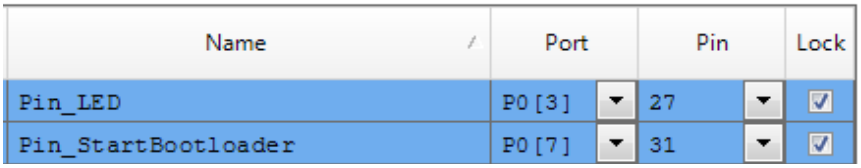


Figure 24　Bootloadable_Blue 项目的顶层设计

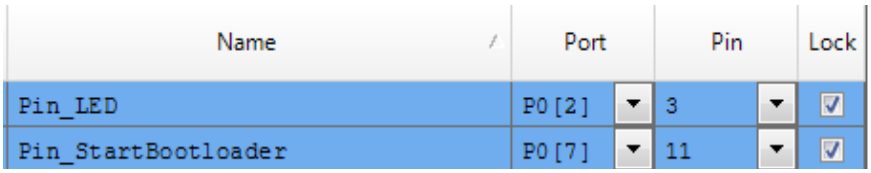
7. 将引脚组件分配给物理引脚。在 **Workspace Explorer** 窗口中，请双击 *Bootloadable_Blue.cydwr* 文件并分配引脚。更多有关引脚分配的信息，请参考英飞凌套件的套件用户指南。**Figure 15** 显示的是 **CY8CKIT-042** 套件电路板的引脚分配，而 **Figure 16** 则显示的是 **CY8CKIT-040** 套件电路板的引脚分配。

项目



Name	Port	Pin	Lock
Pin_LED	P0[3]	27	☑
Pin_StartBootloader	P0[7]	31	☑

Figure 25 CY8CKIT-042 Bootloadable_Blue 项目的引脚分配



Name	Port	Pin	Lock
Pin_LED	P0[2]	3	☑
Pin_StartBootloader	P0[7]	11	☑

Figure 26 CY8CKIT-040 Bootloadable_Blue 项目的引脚分配

8. 编译项目；并生成 ISR 组件 API 文件。然后将代码添加到中断服务子程序内，以设置变量 `bootload_flag`。代码如下所示。

```
CY_ISR(isr_EnterBootloader_Interrupt)
{
      /* Place your Interrupt code here.
      */
      /* `#START
      isr_EnterBootloader_Interrupt`
      */
  bootload_flag=1u;
  Pin_StartBootloader_ClearInterrupt();
        /* `#END` */
}
```

Note: `bootload_flag` 变量被定义在 main.c 内，因此在 isr_EnterBootloader.c 文件内必须声明它是一个外部变量。同时，要避免出现编译警报，请在 isr_EnterBootloader.c 文件中添加`#include <device.h>`。

9. 在此应用笔记的附件中，包含一个已完成的 Bootloadable_Blue 项目。因此，可以将示例代码从这个附件项目的 *main.c* 文件添加到项目的 *main.c* 文件内。

   `main()`函数会持续检查 `bootload_flag` 变量。如果该变量被置 1，`main()`将关闭 LED，并通过调用 API 函数 `Bootloadable_Load()`调用 Bootloader。

10. 再次编译项目。按照**使用一个 PC 主机进行引导加载**一节中所介绍的步骤，下载 Bootloadable 项目 *Bootloadable_Blue.cyacd*。应用程序运行后，目标上的套件 RGB LED 会闪烁蓝色。
11. 按下 **SW2** 开关以进入 Bootloader 后，套件 RGB LED 将开始闪烁红色发光。
12. 启动 Bootloader 主机程序，然后选择其他 Bootloadable 文件 (如 *Bootloadable_Green.cyacd*)，再按下 **Program** 按键。成功加载新的应用程序后，套件 RGB LED 闪烁为绿色。

项目

Note: 要想再次进行引导加载，必须复位器件并快速下载新的.cyacd 文件。这是因为 Bootloadable_Green 应用不能通过调用 `Bootloadable_Load()` 函数来调用 Bootloader，因此只在复位后才能调用 Bootloader。

2.3 I2C Bootloader 主机

除了学习示例项目外，了解 Bootloader 主机项目的常用结构可帮助您编译自己的 Bootloader 主机系统。

2.3.1 Bootloader 主机程序

Figure 27 显示的是一个 Bootloader 系统的协议级框图。Bootloader 主机和目标机各自包括两种模块：一个内核和一个通信层。

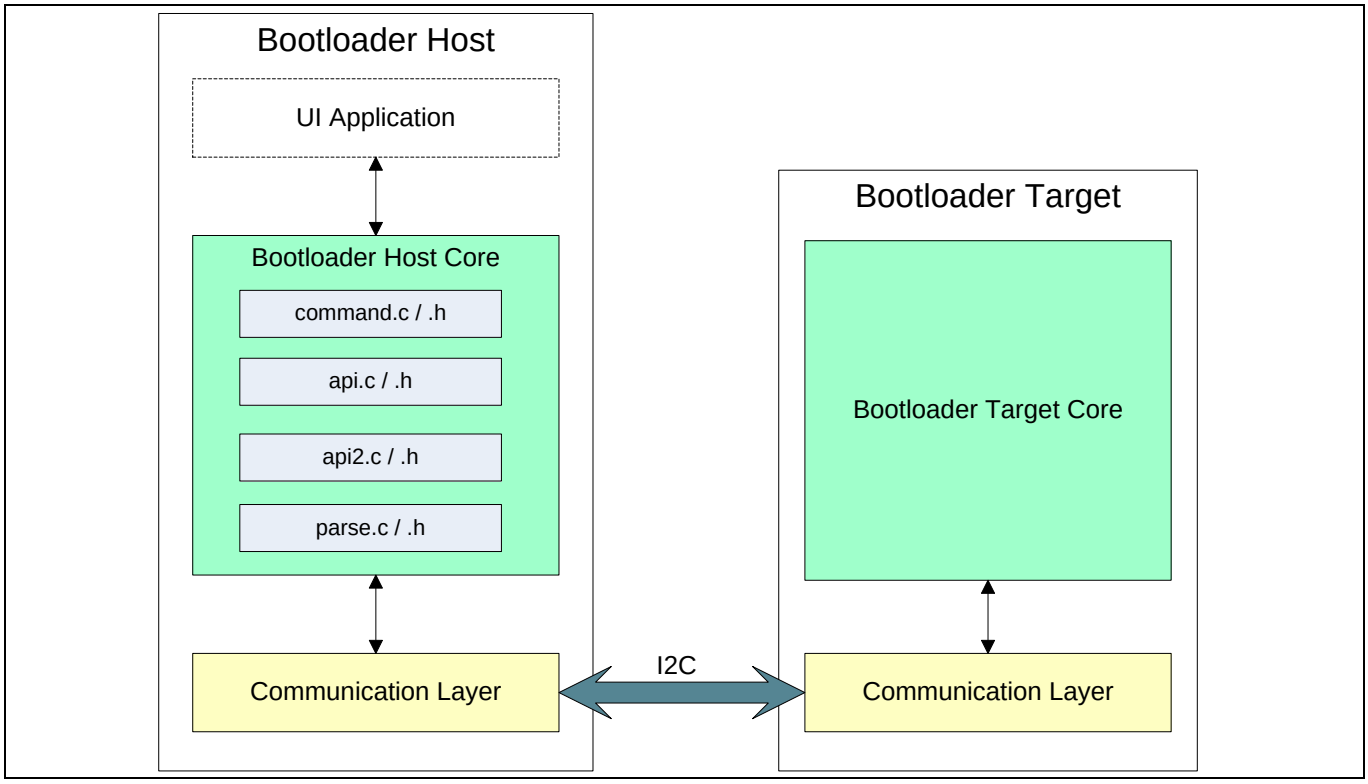


Figure 27 引导加载的协议级框图

Bootloader 主机内核将执行所有引导加载操作。它将指令包和闪存数据发送给目标器件。根据目标的响应，Bootloader 主机内核将决定是否继续进行引导加载。

Bootloader 目标内核先对来自主机的指令进行解码，然后通过调用闪存例程 (如擦除行、编程行和验证行) 执行它们，并生成响应包。

主机和目标上的通信层为引导加载协议提供了物理层支持。它们包含了用于执行该功能的通信协议 (I^2C) 的 API。该层用于在主机和目标之间发送和接收协议包。

项目

2.3.2 Bootloader 系统 API

当您编译某个 Bootloader 项目时，PSoC™ Creator 将自动生成 Bootloader 目标内核和通信层中的所有 API。

PSoC™ Creator 也提供了内核的主机端 API，其路径为：

<install folder> \ PSoC Creator \ 4.2\ PSoC Creator \ cybootloaderutils

更多有关这些 API 文件的信息，请参考**附录 D** 中介绍的内容。

唯一需要您写入的代码是用于通信层的主机端 API 函数，它们位于 *communication_api.c / .h* 内。这四个函数为：`OpenConnection()`、`CloseConnection()`、`ReadData()`以及 `WriteData()`。指向这些函数的指针位于“CyBtldr_CommunicationsData”结构中。该结构被定义在 *cybtldr_api.h* 中。

在该项目中，使用 **Bootloadable_Blue 项目 — 示例 2** 生成*.cyacd* 文件。在按下开关来调用 Bootloader 时，需要使用这些文件。为该项目生成一个*.cyacd* 文件，并将它命名为 *Bootloadable_BlueLED.cyacd*。类似的，通过将 Pin_LED 分配给 P0[2]引脚，可以生成一个 *Bootloadable_GreenLED.cyacd*。该应用笔记提供了这两个文件，供快速参考。

2.3.3 创建 I2C Bootloader 主机项目的步骤

本节介绍了如何使用 PSoC™创建一个嵌入式 I^2C Bootloader 主机项目。在使用该项目时可以引导加载另一个 PSoC™器件。对于该项目，通过轮流按下开关，主机可以引导加载两个不同的 Bootloadable 文件 (*.cyacd* 文件)。

1. 创建新的 PSoC™ Creator 项目。选择目标 PSoC™器件，将该项目命名为 **I2C_Bootloader_Host**，并为该项目创建一个新的工作区。
2. 由于 Bootloader 项目有一个 I^2C 从设备，因此主机项目必须有一个 I^2C 主设备。这样，需要将一个 I^2C (SCB 模式) 组件添加到顶层设计的原理图内。同样，也将数字输入引脚、中断以及 UART (SCB 模式) 组件添加到顶层设计中。根据 **Table 4** 中显示的内容，命名各个组件。

Table 4 I2C_Bootloader_主机项目的组件列表

组件	名称
I2C_1	I2C
Pin_1	Pin_Switch
isr_1	ISR_Switch
UART_1	UART

3. 要配置 I2C 组件，请双击它。将它设置为主模式。默认的数据速率为 100 Kbps。
4. 数字输入引脚 Pin_Switch 用于启动主机中的引导加载操作。当按下套件按键时，该引脚会短路接地，因此需要对它进行配置，使其电阻上拉，并在下降沿生成中断。将 ISR_Switch 组件连接到该引脚的中断输出 (irq)。
5. UART 是用来将引导加载状态或错误代码传输到 PC 的。保留各参数的默认设置 (波特率：115200 kbps，数据位：8 位，奇偶校验：无，停止位：1 位)。

   为按键添加一个注释组件后，该项目顶层设计的具体情况应该与 **Figure 28** 所示的相似。

项目

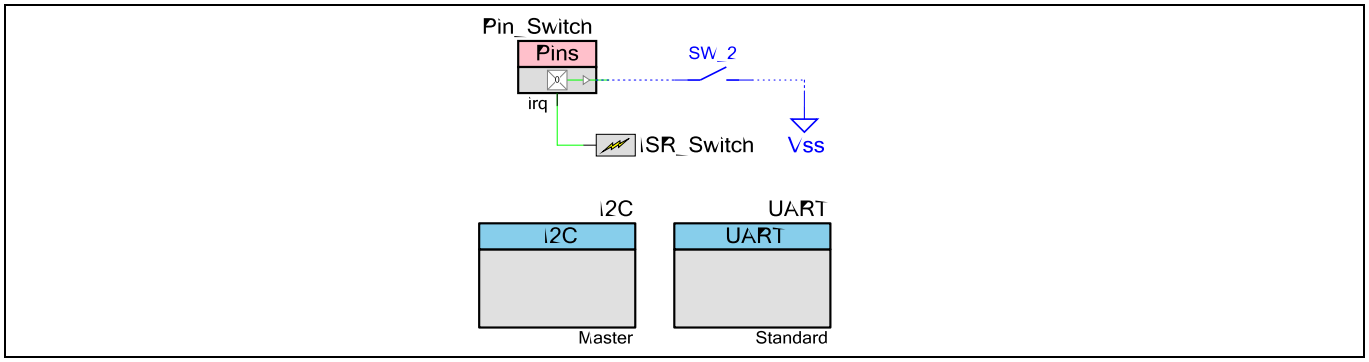
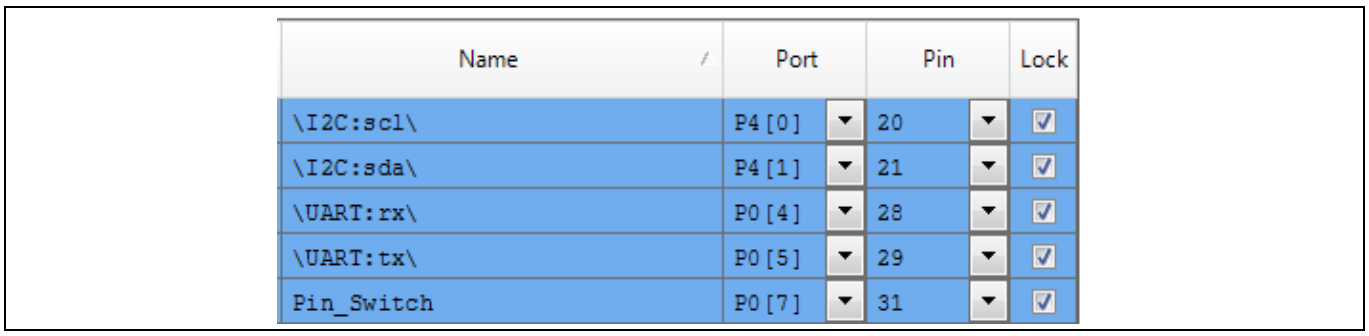


Figure 28 I2C_Bootloader_Host 项目的顶层设计

6. 对输入和输出引脚进行分配。在 **Workspace Explorer** 窗口中，双击 *I2C_Bootloader_Host.cydwr* 文件并分配各引脚。更多有关引脚分配的信息，请参考英飞凌套件的器件数据手册或套件用户指南。

   Figure 29 显示的是 **CY8CKIT-042** 的引脚分配。



Name	Port	Pin	Lock
\I2C:scl\	P4[0]	20	☑
\I2C:sda\	P4[1]	21	☑
\UART:rx\	P0[4]	28	☑
\UART:tx\	P0[5]	29	☑
Pin_Switch	P0[7]	31	☑

Figure 29 I2C_Bootloader_Host 项目的引脚分配

7. 通过编译项目生成 ISR 组件 API 文件。将代码添加到中断服务子程序内，以设置变量 `switch_flag`。代码如下所示。

```
CY_ISR(ISR_Switch_Interrupt)
{
/* Place your Interrupt code here.
 */
/* `#START ISR_Switch_Interrupt` */
  switch_flag = 1u;
Pin_Switch_ClearInterrupt();
/* `#END` */
}
```

Note: `switch_flag` 被定义在 main.c 内，因此在 ISR_Switch.c 文件内必须声明它是一个外部变量。同时，要避免出现编译警报，请在 ISR_Switch.c 文件内添加`#include <device.h>`。

项目

8. 为该项目添加固件。该应用笔记中包含了 I2C_Bootloader_Host 项目。因此，可以将示例代码从这个附件项目的 *main.c* 文件添加到项目的 *main.c* 文件内。

   main.c 文件中的 `main()`函数持续检查 `switch_flag` 变量。该变量为 1 时，将启动引导加载过程。通过 *main.c* 文件中的函数，可以调用 *device.h* 中所定义的 `BootloadStringImage()`。该函数通过使用 Bootloader 主机 API 文件 (主机内核；请参考 **Figure 27**)，可以引导加载*.cyacd* 文件。

   `main()`函数中有另一个变量，它的名称为“`toggle_appcode`.” 每次按下按键时，该变量会在‘0’和‘1’之间交替变换。这样可以使主机交替选择 Bootloadable 文件。

9. 如前面所述，一个 Bootloader 主机内核是根据四个 API 文件编译的。这些文件会执行主机的所有引导加载操作。因此，您的项目中必须包含这些文件。可以在下面目录中查找这些 API 文件：

   <install folder> \ PSoC Creator \ 4.2\ PSoC Creator \ cybootloaderutils

   要添加这些文件，请打开 **Workspace Explorer** 窗口，右键点击项目名称，然后依次选择 **Add > Existing Item**，如 **Figure 30** 所示。添加由 PSoC™ Creator 提供的下列各文件：*cybtldr_api.c / .h*、*cybtldr_command.c / .h*、*cybtldr_parse.c / .h* 以及 *cybtldr_utils.h*。

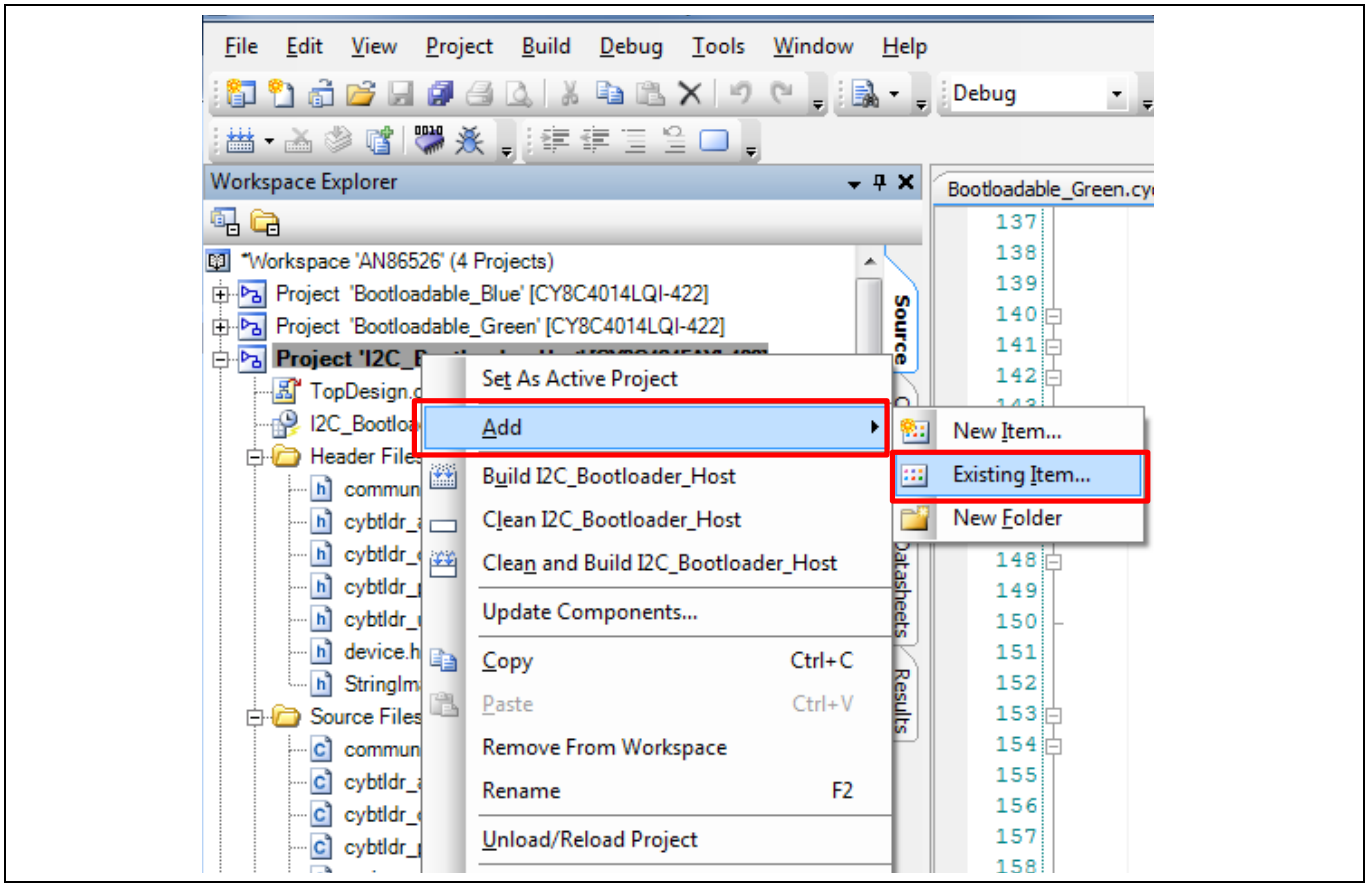


Figure 30　添加 API 文件

10. 除了引导加载 API 文件外，主机还需要通信层的支持。通过添加 *communication_api.c / .h* 文件，可以获得所需支持。您可以从本应用笔记所附带的 I2C_Bootloader_Host 项目内复制这些文件内容 (请根据前面的步骤将这些文件添加到目标中)。通过复制本应用笔记所附带的项目内容，可以更新这些文件。
11. 将 Bootloadable 文件添加到主机系统内。编译好 Bootloadable 文件后，会生成一个*.cyacd* 文件；该文件与*.hex* 输出文件相同。更多有关*.cyacd* 文件的信息，请参考**附录 B**。

项目

以字符串数组的形式复制该文件的内容，其中，每一行为该数组的一个元素。由于有两个 Bootloadable 文件，所以必须定义与其相应的两个数组，即“StringImage_0”和“StringImage_1”。对于每个数组，定义一个宏用于存储该数组中的行数。需要在名称为 *StringImage.h* 的单独文件内定义这些数组 (定义各个字符串前，必须将该文件添加到项目中)。

请参考与本应用笔记相关联的 I2C_Bootloader_Host 项目中的 *StringImage.h* 文件。

编译该项目，并将它编程到英飞凌套件中的 PSoC™内。

3 测试项目

*Note: **CY8CKIT-042** 套件和 **CY8CKIT-040** 套件上都有 RGB LED，如 **Figure 31** 和 **Figure 32** 所示。在这里，绿色 LED 和蓝色 LED 分别用于指示目前哪个 Bootloadable 项目正在运行。*

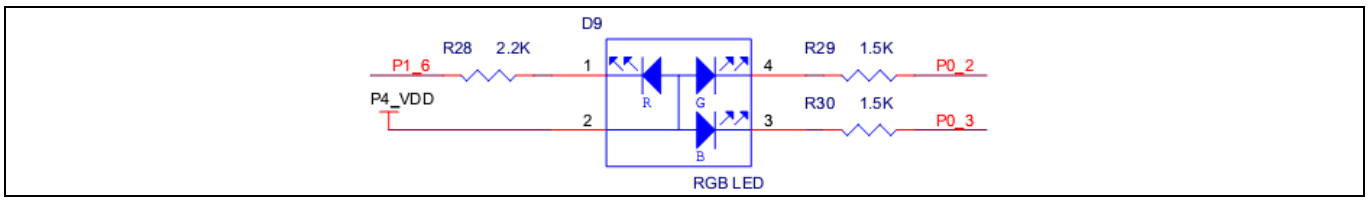


Figure 31 CY8CKIT-042 RGB LED 原理图

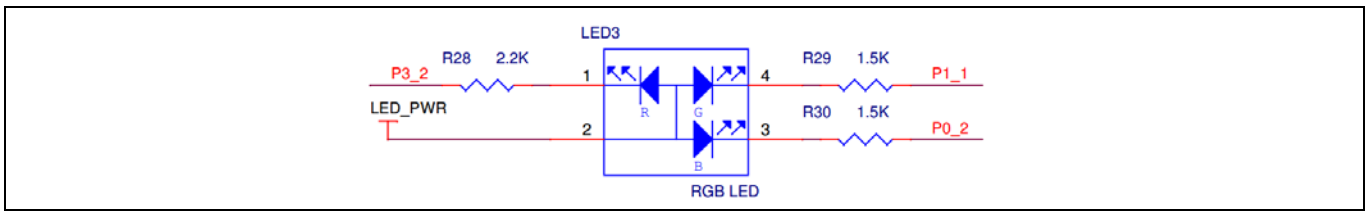


Figure 32 CY8CKIT-040 RGB LED 原理图

3.1 配置套件

对于目标 PSoC™套件，请按照以下步骤进行操作：

1. 使用 USB 线缆 (通过它提供电源) 将目标连接到 PC。
2. 通过 I2C_Bootloader_Red 项目编程目标 PSoC™。

对于主机 PSoC™套件，请按照下述其他步骤进行操作：

1. 使用 USB 线缆 (通过它提供电源) 将主机连接到 PC。
2. 通过 I2C_Bootloader_Host 项目编程主机 PSoC™。
3. 打开任何串行端口查看器 (如 PC 上的 Tera Term)，以显示引导加载信息。

例如，对于 **CY8CKIT-042** 主机套件，将 PSoC™ 4 中分配的 Rx (P0[4]) 和 Tx (P0[5]) 引脚连接到扩展插头 J8 上的引脚 10 和引脚 9 上。更多有关信息，请参考 **CY8CKIT-042 PSoC™ 4 Pioneer 套件指南**。使用 I2C_Bootloader_Host 项目对该套件进行编程并打开串行端口查看器，从而显示引导加载信息。

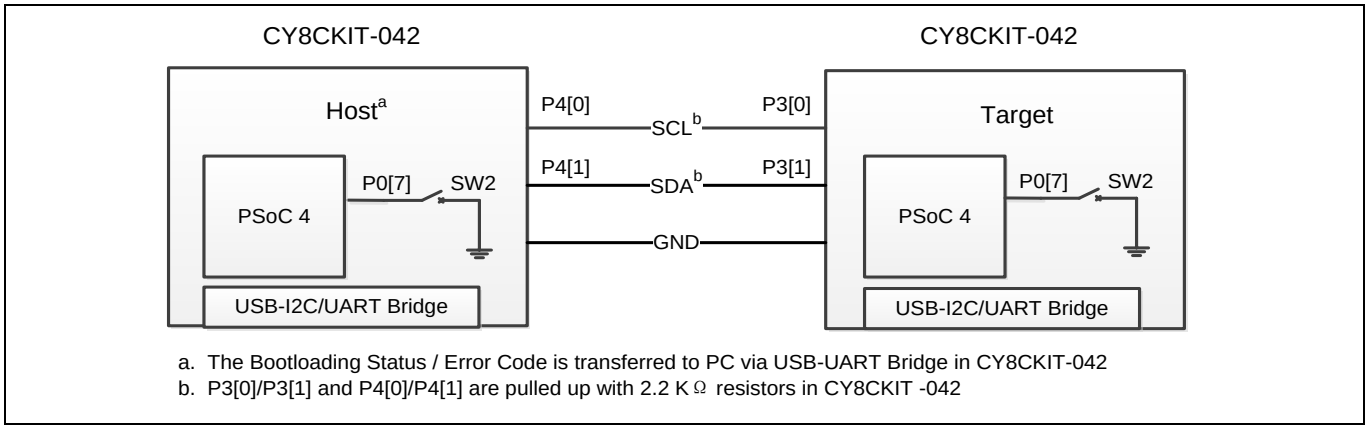
要想将主机套件连接到目标器件上，请按照下列步骤进行操作：

1. 将主机套件的 SCL 和 SDA 引脚连接至目标套件的相应 SCL 和 SDA 引脚上。
2. 同时短路该两个套件的接地引脚。

例如，对于 **CY8CKIT-042** 主机和 **CY8CKIT-042** 目标，将主机的 P4[0] (SCL) 和 P4[1] (SDA) 连接到目标的相应 P3[0] (SCL) 和 P3[1] (SDA) 引脚，如 **Figure 33** 所示。对于 **CY8CKIT-042** 主机和 **CY8CKIT-040** 目标，则将主机的 P4[0] (SCL) 和 P4[1] (SDA) 连接至目标的相应 P1[2] (SCL) 和 P1[3] (SDA) 引脚，如 **Figure 34** 所示。同时短路该两个套件的接地引脚。

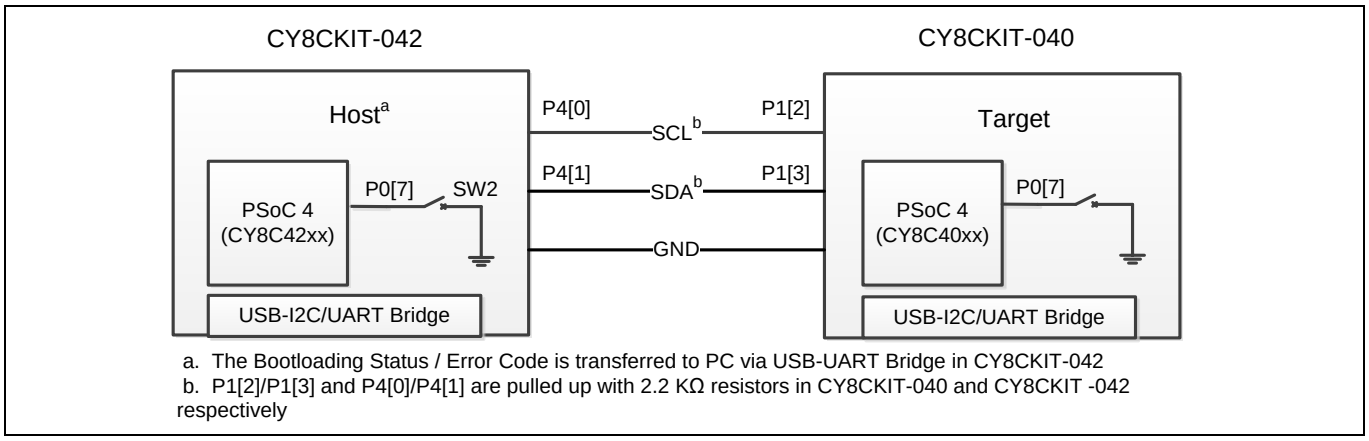
Figure 33 和 **Figure 34** 显示的是这些连接。

测试项目



a. The Bootloading Status / Error Code is transferred to PC via USB-UART Bridge in CY8CKIT-042
b. P3[0]/P3[1] and P4[0]/P4[1] are pulled up with 2.2 KΩ resistors in CY8CKIT -042

Figure 33 主机/目标连接 (CY8CKIT-042 目标)



a. The Bootloading Status / Error Code is transferred to PC via USB-UART Bridge in CY8CKIT-042
b. P1[2]/P1[3] and P4[0]/P4[1] are pulled up with 2.2 KΩ resistors in CY8CKIT-040 and CY8CKIT -042 respectively

Figure 34 主机/目标连接 (CY8CKIT-040 目标)

3.2 验证结果

配置完套件后，您可以根据下面的介绍测试示例项目：

- 第一次按下**主机**套件上的按键 (CY8CKIT-042 的 P0[7]) 时，Bootloadable_*GreenLED.cyacd* 文件将被加载到目标 PSoC™内。成功完成后，串行端口查看器上将显示“Bootloaded. LED blinks with green color on Target”信息，并且目标机 LED 闪烁绿色发光。
- 对于后续的引导加载操作，请点击目标套件上的按键 (**CY8CKIT-042** 的 P0[7])。当检测到有按键被按下时，PSoC™将进入 Bootloader。PSoC™ 4 正在执行 Bootloader 时，红色 LED 将闪烁发光。
- 再次按下主机套件上的按键时，*Bootloadable_BlueLED.cyacd* 文件将被引导加载到目标 PSoC™内。引导加载成功后，串行端口查看器上将显示“Bootloaded. Blue LED blinks on the target”信息，并且目标机 LED 闪烁蓝色发光。

4 总结

该应用笔记解释了如何使用 I^2C 通信接口来引导加载 PSoC™。同时也介绍了 Bootloader 主机的基本编译模块，以及如何编译一个嵌入式 I^2C Bootloader 主机。

Bootloader 是一种现场升级的标准方法。通过 PSoC™ Creator 进行全部配置，您可以轻松地为 PSoC™添加一个 Bootloader。

如需更多高级信息，请参见**附录**部分以及《**PSoC™ 4** 技术参考手册》中的内容。

5 相关应用笔记

为更好地理解 Bootloader 和闪存编程，请参考下面各应用笔记：

- **AN73854** — PSoC™ 3、PSoC™ 4 以及 PSoC™ 5LP 中 Bootloader 的简介
- **AN60317** — PSoC™ 3 和 PSoC™ 5LP I^2C Bootloader
- **AN68272** — PSoC™ 3、PSoC™ 4 以及 PSoC™ 5LP UART Bootloader
- **AN84858** — 使用外部微控制器 (HSSP) 对 PSoC™ 4 进行编程
- **AN61290** — PSoC™ 3 和 PSoC™ 5LP 硬件设计注意事项
- **AN79953** — PSoC™ 4 入门手册

要想了解有关 PSoC™ 4 的其它特性和功能，请点击**此处**，查找应用笔记的完整列表。

6 相关项目

Table 5 显示的是随本应用手册附上的项目列表。

Table 5 本应用笔记附带的项目

设计项目的名称	说明
I2C_Bootloader_Red	该项目演示了如何使用 PSoC™ Creator 为 PSoC™创建一个 I2C Bootloader 项目。该 Bootloader 驱动 CY8CKIT-042*套件上的红色 LED 闪烁。
Bootloadable_Green	该项目演示了如何使用 PSoC™ Creator 为 PSoC 创建 Bootloadable 项目。该应用程序驱动 CY8CKIT-042*套件上的绿色 LED 闪烁。
Bootloadable_Blue	该项目演示了如何创建 Bootloadable 项目，同时介绍了从 Bootloadable 项目进入 Bootloader 的方法。该应用程序驱动 CY8CKIT-042*套件上的蓝色 LED 闪烁。
I2C_Bootloader_Host	这是一个 Bootloader 主机示例程序，用于演示一个 PSoC™引导加载另一个 PSoC™。

* 这些项目可以很容易地适应其他套件。

7 PSoC™资源

在英飞凌网站 **www.cypress.com** 上提供了大量数据，有助您正确选择 PSoC™器件来进行设计，从而使您能够快速并有效地将器件集成到设计中。有关资源的完整列表，请参考 **KBA86521 — 如何使用 PSoC™ 3、PSoC™ 4 和 PSoC™ 5LP 的资源进行设计**。下面提供了 PSoC™ 4 的简要列表：

- **概况：PSoC™产品系列, PSoC™产品路线图**
- **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：**描述并提供了适用于 **PSoC™ 4000**, **PSoC™ 4000S**, **PSoC™ 4100**, **PSoC™ 4100S**, PSoC™ 4100PS, PSoC™ 4100S Plus, **PSoC™ 4200**, **PSoC™ 4xx7 BLE**, **PSoC™ 4200-M**, **PSoC™ 4200-L** 器件系列的电气规范。
- **CapSense 设计指南：**了解如何在 PSoC™ 4 器件系列中设计电容式触摸感应应用。
- **应用笔记和代码示例**包括了从基本到高级的广泛主题。许多应用笔记包括了代码示例。PSoC™ Creator 提供了其他代码示例，请参考**代码示例**。
- **技术参考手册 (TRM)** 对每个 PSoC™ 4 器件系列中所用的架构和寄存器进行了详细说明。
- **开发套件：**
  - **CY8CKIT-040**, **CY8CKIT-041**, **CY8CKIT-042**, **CY8CKIT-042-BLE**, **CY8CKIT-044** 和 **CY8CKIT-046** PSoC™ 4 套件均为易用且廉价的开发平台。这些套件包括用于连接 Arduino™兼容扩展板和 Digilent® Pmod™子卡的连接器。
  - **CY8CKIT-001** 是所有 PSoC™器件系列经常使用的开发平台。
  - **CY8CKIT-147** and **CY8CKIT-149** 是 PSoC™ 4 器件非常低价的选样原型开发平台。
- **MiniProg3** 器件提供一个用于进行闪存编程和调试的接口。

7.1 PSoC™ Creator

PSoC™ Creator 是一个基于 Windows 的免费集成开发环境 (IDE)。通过它可以同时对 PSoC™ 3, PSoC™ 4 和 PSoC™ 5LP 器件进行硬件和固件设计 (请参考 **Figure 35**)。使用 PSoC™ Creator，可以执行以下操作：

1. 将组件图标拖放到主要设计工作区中，以进行您的硬件系统设计。
2. 对您的应用固件和 PSoC™硬件进行协同设计。
3. 使用配置工具配置各组件。
4. 了解包含一百多个组件的库。
5. 查看组件数据手册。

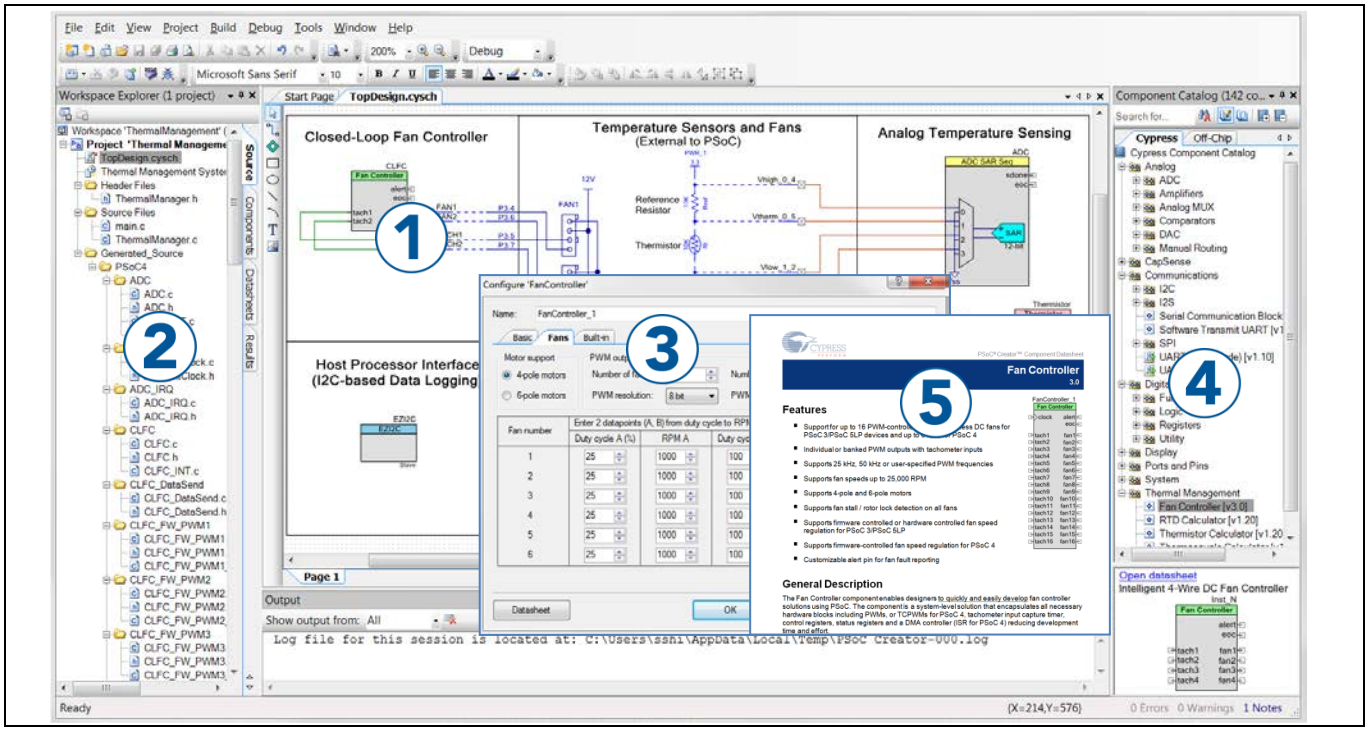


Figure 35 PSoC™ Creator 特性

7.2 代码示例

PSoC™ Creator 包含了多个代码示例项目。可以从 PSoC™ Creator 的起始页上获取这些项目，如 **Figure 36** 所示。

这些示例项目为您提供完整的设计 (并非一个空白页)，从而可以加快您的设计过程。示例项目还介绍了如何将 PSoC™ Creator 组件使用于不同应用中。此外，它还包含了多个代码示例和数据手册，如 **Figure 36** 所示。

在 **Figure 37** 显示的 Find Example Project (查找示例项目) 对话框中现在有几个选项：

- 根据架构或器件系列 (例如：PSoC™ 3, PSoC™ 4 或 PSoC™ 5LP)、类型或关键词等选项对示例进行筛选。
- 从 **Filter Options** (滤波选项) 的示例菜单中选择。
- 通过 **Documentation** (文档) 选项卡，查看选出的数据手册。
- 查看所选的代码示例。您可以复制该窗口中的代码然后将其粘贴到您的项目内，从而加快代码的开发过程，或
- 根据已选项目创建一个新的项目 (若需要可添加新的工作区)。通过为您提供一个完整的基本设计，它可以加快您的设计过程。然后，您可以根据自己的应用来调整该设计。

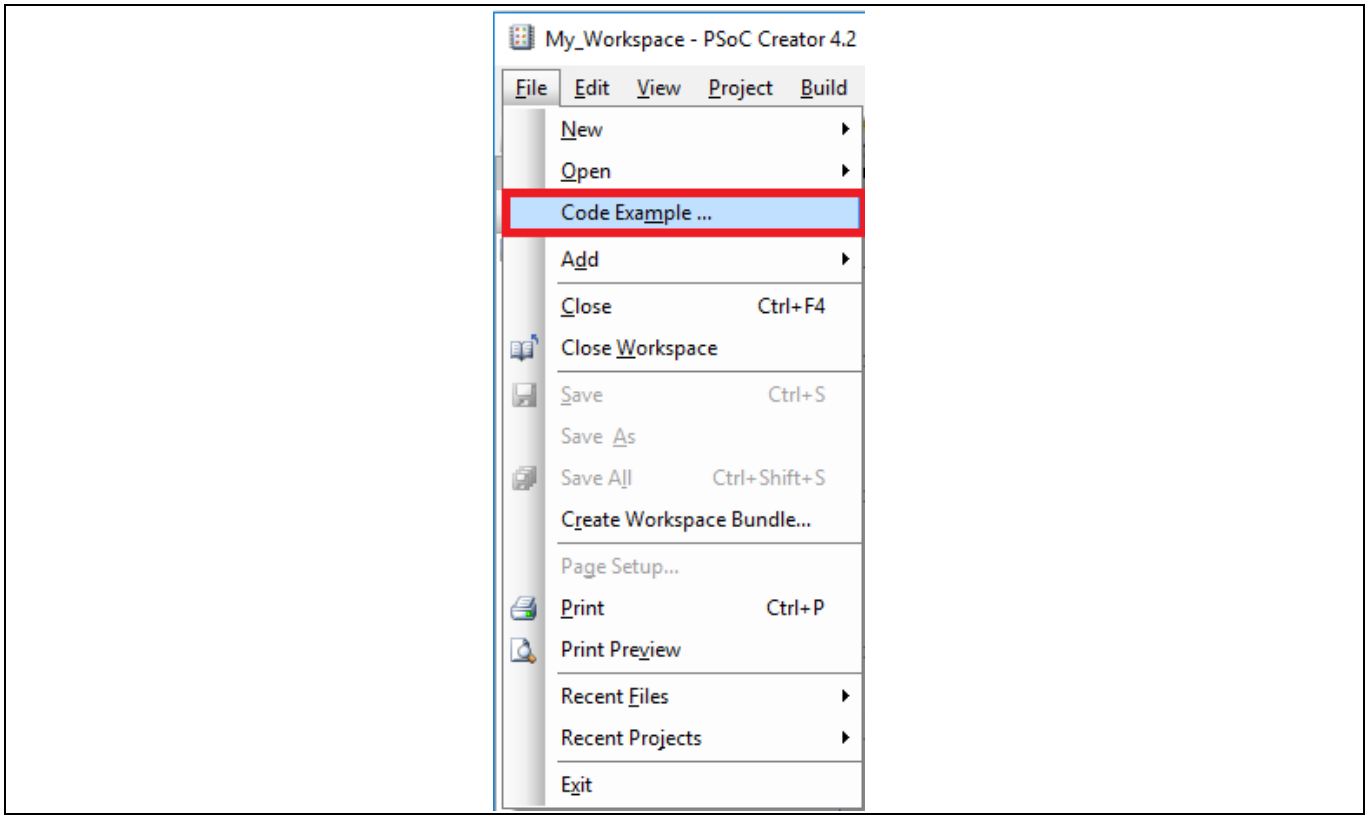


Figure 36 PSoC™ Creator 中的代码示例

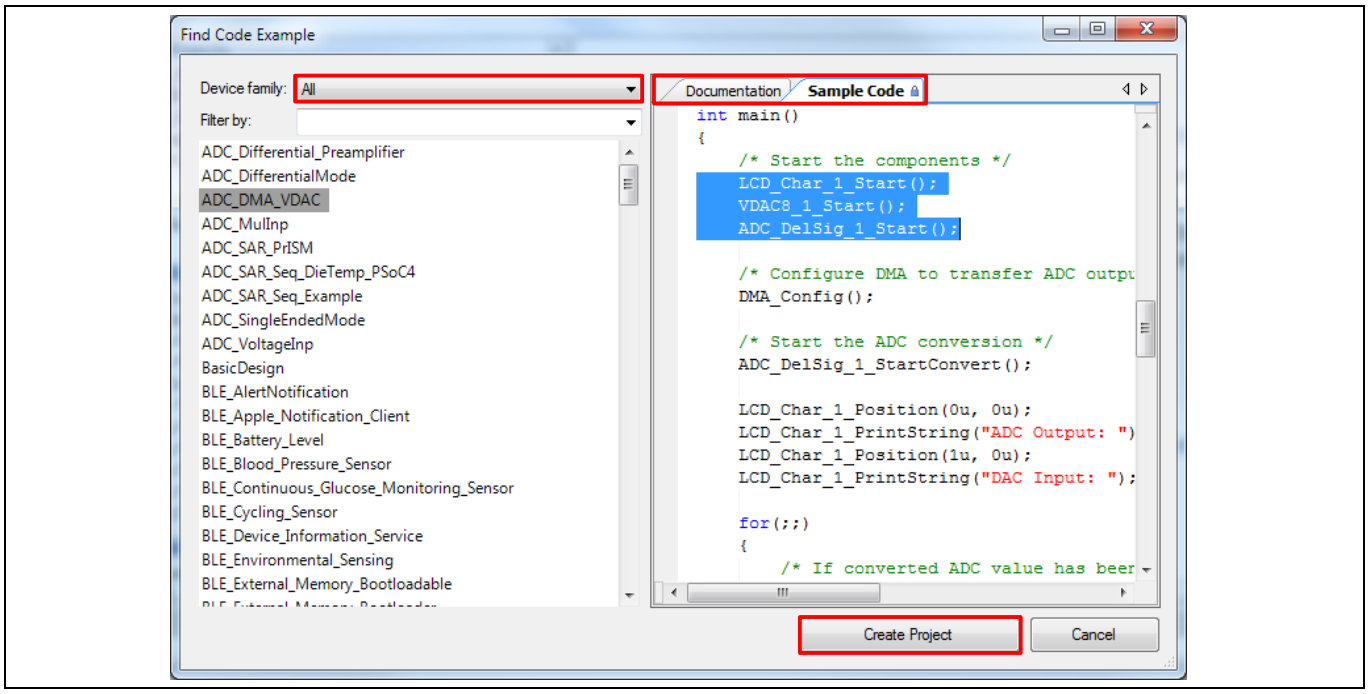


Figure 37 带样本代码的代码示例项目

7.3 PSoC™ Creator 帮助

请访问 **PSoC™ Creator 主页**以下载 PSoC™ Creator 的最新版本。启动 PSoC™ Creator，并导航到下列各项：

- **快速入门指南：**依次选择 **Help** > **Documentation** > **Quick Start Guide**。本指南提供了开发 PSoC™ Creator 项目的基本知识。
- **简单的组件示例项目：**依次选择 **File** > **Open** > **Example projects**。这些示例项目展示了如何配置及使用 PSoC™ Creator 组件。
- **入门设计：**依次选择 **File** > **New** > **Project** >**PSoC 4 Starter Designs**。这些入门设计展示了 PSoC™ 4 的独特特性。
- **系统参考指南：**请依次选择 **Help** > **System Reference** > **System Reference Guide**。该指南列出并描述了 PSoC™ Creator 提供的系统功能。
- **组件数据手册**：右击组件，并选择“Open Datasheet”项。请访问 **PSoC™ 4 组件的数据手册**网页，获取所有 PSoC™ 4 组件的数据手册列表。
- **文档管理工具**：PSoC™ Creator 提供了文档管理工具，有助您查找和查看文件资源。要想打开文档管理工具，请选择菜单项 **Help** >**Document Manager**。

7.4 技术支持

若有任何疑问，我们的技术支持团队很乐意为您提供帮助。您可以在**赛普拉斯技术支持**页面上创建一个技术支持请求。

如果您在美国，可以通过拨打我们的免费电话，直接与技术支持团队联系：+1-800-541-4736。选择提示符处的第 8 项。

若想快速获得支持，您同样可以使用下面的支持资源。

- **自助**
- **所在地销售办事处**

8 附录 A：存储器

8.1 闪存存储器的详细信息

闪存存储器用来存储固件、批量数据、器件配置数据、工厂配置数据以及用户定义的闪存保护数据。**Figure 38** 显示的是 PSoC™中闪存存储器的物理结构。

PSoC™闪存存储器分为不同模块，这些模块被称为“阵列”。阵列 ID 是用来识别阵列的唯一方法。每个阵列占用闪存存储器的 128 或 256 行。每一行具有用于 PSoC™ 4100、4200、4000S 和 4100PS 系列器件的 128 个数据字节，以及用于 4100S 器件系列的 256 个数据字节。在 4000 器件系列中，每行的大小为 64 个数据字节。因此，每个阵列可以存储 8 KB、16 KB、32 KB 或 64 KB 的指令和数据。

PSoC™ 4100、4200、4000S 和 4100PS 系列器件的最大闪存空间为 32 KB，因此它只有一个阵列，并且唯一有效的阵列 ID 为 0。PSoC™ 4000S 器件的最大闪存空间为 64 KB，所以它只有一个阵列，并且唯一有效的阵列 ID 为 0。PSoC™ 4100S Plus 器件的最大闪存空间为 128 KB，并且唯一有效的阵列 ID 为 0。PSoC™ 4000 器件的最大闪存空间为 16 KB，所以它只有一个阵列，并且唯一有效的阵列 ID 为 0。

每次只能编程闪存存储器中的一行。可以擦除闪存中的某一行，也可以擦除整个闪存。每个行的唯一识别方法是阵列 ID 和行编号的组合。

Figure 38 也显示了 Bootloader 占据闪存中前 X 行的情况。X 设置为使以下内容具有足够的空间：

- Bootloader 的向量表，其起始地址为 0
- Bootloader 项目的配置字节
- Bootloader 项目的代码和数据
- 闪存中 Bootloader 部分的校验和

对于 PSoC™，该向量表包含了 Bootloader 项目的初始堆栈指针 (SP) 值以及 Bootloader 项目代码的起始地址。另外它还包含了用于 Bootloader 的异常向量及中断向量。

Bootloadable 项目紧挨着 Bootloader，占用从首个 128 字节边界开始的闪存。闪存的区域包括：

- Bootloadable 项目的向量表
- Bootloadable 项目的代码和数据

闪存的最高 64 字节块用作两种项目的公共区域。此模块中所保存的参数包括：

- Bootloadable 项目在闪存中的入口地址 (4 字节地址)
- Bootloadable 项目占据的闪存大小 (闪存行的数量)
- 闪存 Bootloadable 部分的校验和 (一个字节)
- 闪存 Bootloadable 部分的字节大小 (4 个字节)。

更多有关闪存存储器中元数据布局的信息，请参见**闪存的元数据布局**。

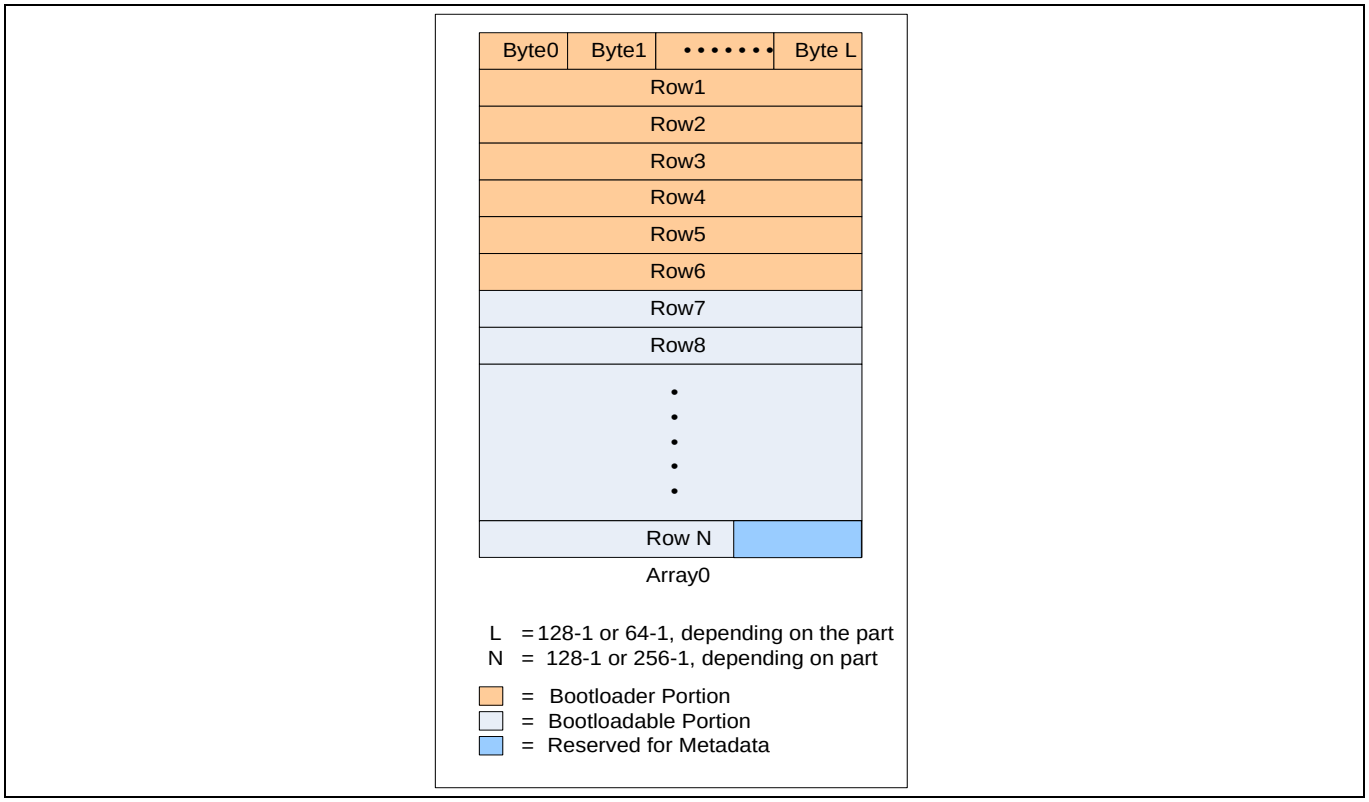


Figure 38 PSoC™中闪存存储器的物理结构

8.2 PSoC™中存储器的使用情况

Bootloader 项目有两种类型：标准 Bootloader 和多应用 Bootloader。对于要保证始终有可运行的有效应用的设计，多应用 Bootloader 是非常有用的。不过，此功能有明显的限制，即为每个应用只能占用有效闪存存储器的一半。

Figure 39 显示的是每种 PSoC™ Creator 项目中闪存存储器的使用情况。

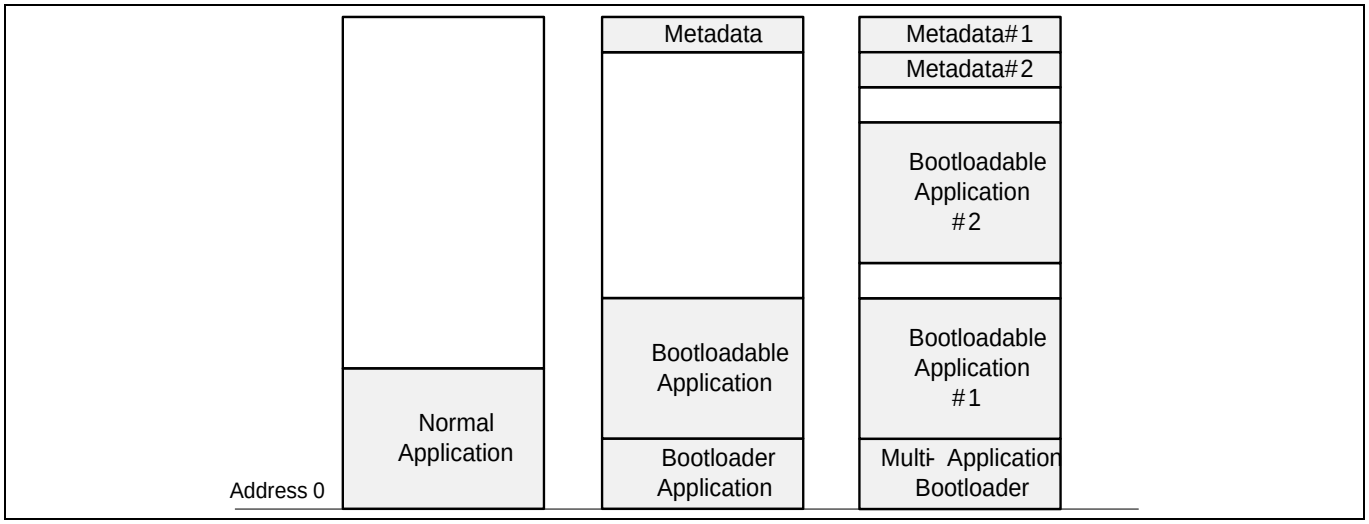


Figure 39 闪存存储器的使用情况

8.3 内存中的元数据布局

元数据部分是闪存中最高 64 字节块，作为 Bootloader 项目和 Bootloadable 项目的公共区域使用，如 **Figure 39** 所示。该块存储了各种参数，如 **Table 6** 所示。多应用 Bootloader 有两个元数据区域。

Table 6 元数据布局

地址	PSoC™
0x00	应用校验和
0x01	应用地址
0x02	
0x03	
0x04	
0x05	Bootloader 的最后一行
0x06	
0x07	
0x08	
0x09	应用程序大小
0x0A	
0x0B	
0x0C	
0x0D	NA
0x0E	NA
0x0F	NA
0x10	应用程序活动状态
0x11	校验类型
0x12	Bootloader 程序版本
0x13	
0x14	Bootloadable 应用 ID
0x15	
0x16	Bootloadable 应用版本
0x17	
0x18	Bootloadable 应用 自定义 ID
0x19	
0x1A	
0x1B	
0x1C-0x3F	NA

*Note: 对于多应用 Bootloader，第 2 个 Bootloadable 的元数据的 Bootloader 最后一行 (**Figure 2**) 是闪存部分中第 1 个 Bootloadable 的最后一行，不是 Bootloader 的最后一行。*

8.3.1 闪存保护

如果 Bootloader 的代码无效，便不能使用该产品。因此，需要防止不小心将闪存内 Bootloader 部分覆盖掉。

PSoC™的保护包括芯片级保护 (“打开”，“受保护”和“Kill”) 和闪存级保护。更多有关信息，请参见器件数据手册或技术参考手册 (TRM)。通过闪存的保护性能可防止对专有代码进行复制或逆向项目。该性能还可以防止不小心对闪存内 Bootloader 部分进行写操作。

闪存级保护有两种，如 **Table 7** 所示。对闪存的每一行进行配置，使之有不同的保护级别。使用 PSoC™ Creator (*.cydwr* 文件的 Flash Security 选项卡) 设置该保护级别。

Table 7 PSoC™的闪存保护级别

保护级别	支持	不支持
无保护	外部读写访问 内部读写访问	–
完全保护	内部读访问 外部读访问	外部写访问 内部写访问

闪存内 Bootloader 部分的保护级别被配置为完全保护后，便不能修改该 Bootloader。更改保护级别或修改 Bootloader 代码的唯一方法是全部擦除闪存，然后使用 SWD 接口重新对其进行编程。

闪存 Bootloader 保护的示例如下。

8.3.2 闪存保护的示例

编译 Bootloader 项目后，PSoC™ Creator Output 窗口将显示所使用的闪存容量。例如，如果 I2C_Bootloader 项目占用了 4352 个字节的闪存，那么，输出将为 (对于带有 32 B 闪存的 PSoC™)：

```
Flash used: 4352 of 32768 bytes (13.3%).
```

因此，Bootloader 占用闪存内的 34 行 (4352 /128)，即 0x0000 到 0x10FF 的闪存空间。将这些行的闪存保护级别配置为 Full Protection (PSoC™ Creator 中.*cydwr* 文件的 Flash Security 选项卡)。可以将其余各行的保护级别配置为无保护 (默认)，如 **Figure 40** 所示。更多有关如何使用 Flash Protection 对话框的信息，请参见 PSoC™ Creator 帮助文章中闪存安全的内容。

附录 A：存储器

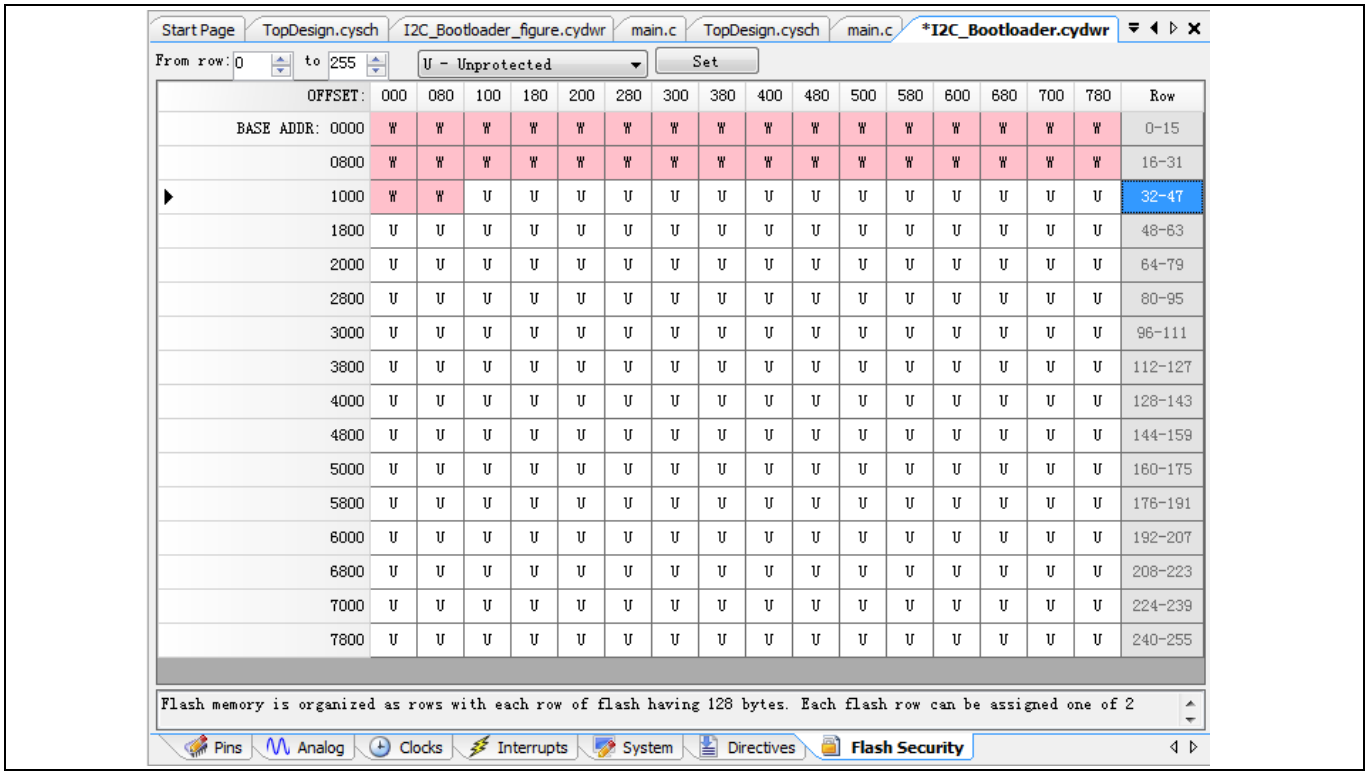


Figure 40 PSoC™ Creator 中的闪存保护

9 附录 B：项目文件

9.1 Bootloadable 输出文件

每当编译了某个 PSoC™ Creator 项目后，都会生成一个*.hex* 类的输出文件。进行编程时，可使用 SWD 接口将该文件编程到 PSoC™。

对于 Bootloadable 项目，*.hex* 文件是 Bootloadable 项目和相关 Bootloader 项目的组合*.hex* 文件。通常在生产环境中通过 SWD 接口将这两个项目编程至 PSoC™闪存。

9.1.1 *.cyacd 文件格式

当编译 Bootloadable 项目时，还会生成一个*.cyacd* 类的附加文件 (应用代码和数据)。该文件包含一个文件头，紧跟着是若干闪存数据行。除头文件以外，该文件中的每一行均表示闪存数据的一整行。数据以高位优先的格式存储为 ASCII 数据。因此，在引导加载中，必须解析该文件的内容 (从 ASCII 转换到 hex)。对于编程*.hex* 类型的文件，解析是不需要的。

文件头的格式如下：

```
[4 bytes Silicon ID] [1 byte Silicon rev] [1 byte checksum type]
```

闪存行的格式如下：

```
[1 byte array ID] [2 bytes row number] [2 bytes data length] [N bytes of
data] [1 byte checksum]
```

文件头中的校验和类型指示在引导加载 Bootloader 和 Bootloader 主机之间发送的数据包的校验和类型。如果该字节为 0，校验和为基本求和。如果该字节为 1，校验和为 CRC-16。

10 附录 C：主机/目标通信

10.1 通信流程

Bootloader 功能流程一节描述了 PSoC™中 Bootloader 的操作，另外 **I2C Bootloader 主机**一节介绍了 Bootloader 主机的构建模块。在这个背景下，**Figure 41** 解释了引导加载期间主机和目标之间的通信流程。同时介绍了通信顺序，包括将指令发送到目标并接收响应。要想了解引导加载指令的完整列表及其代码和预期响应，请参见**指令和状态/错误代码**。

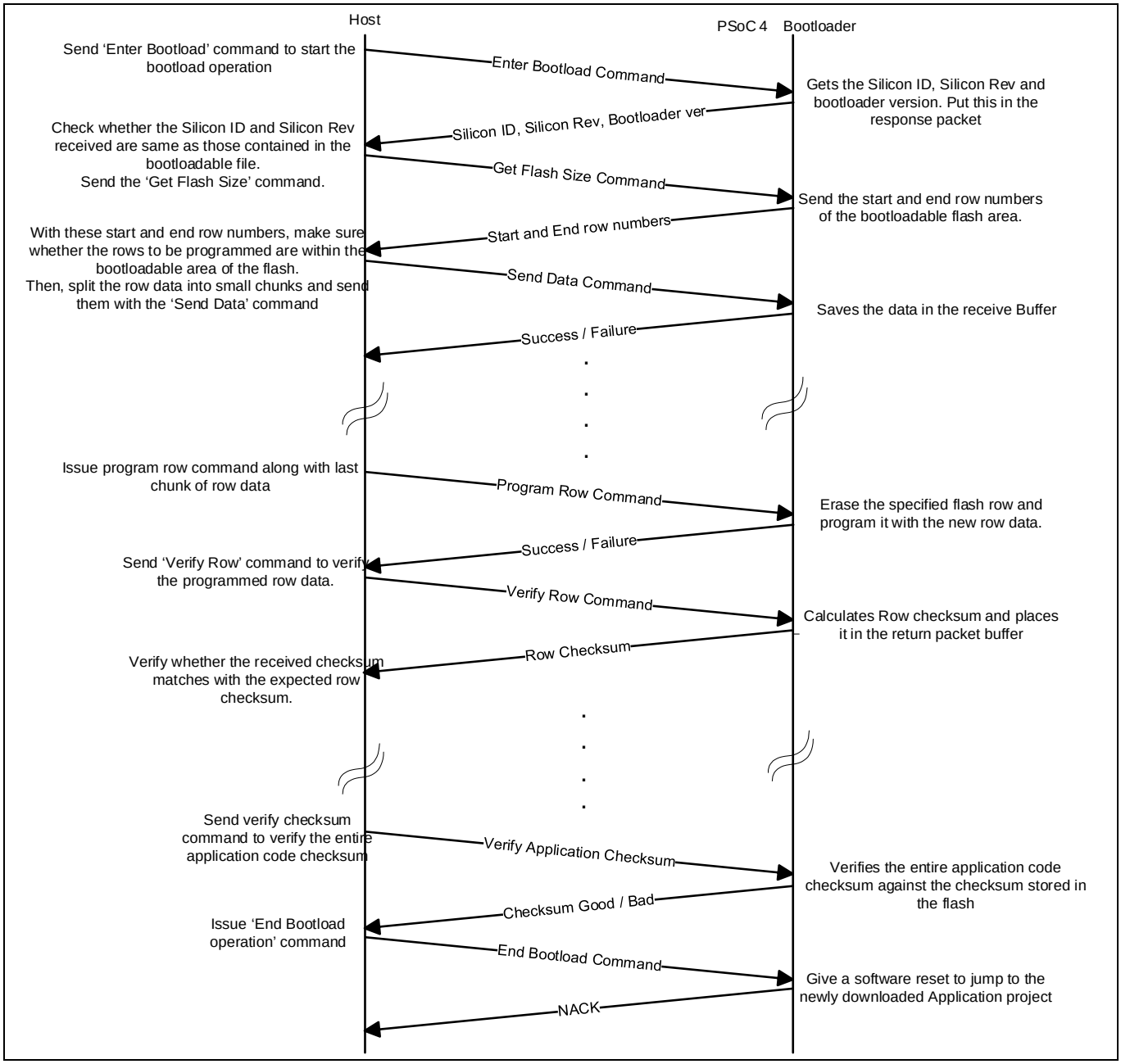


Figure 41 引导加载过程中的通信流程

10.2 协议数据包格式

引导加载操作涉及到主机和目标之间的指令及响应数据包的交换。这些数据包有特定的格式，如 **Figure 42** 所示。

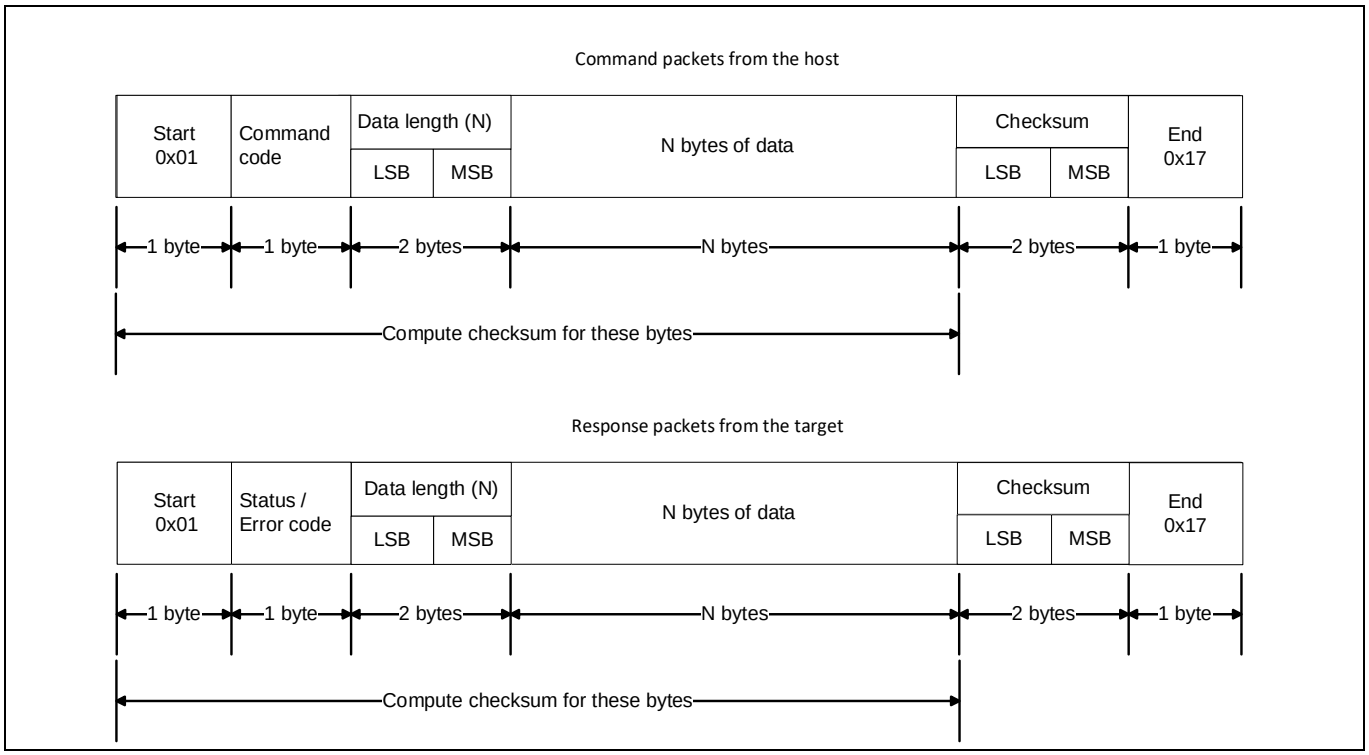


Figure 42 引导加载的数据包格式

每个数据包都包含校验和字节。根据 Bootloader 项目的设置，该校验和将是基本求和 (2 的补吗) 或 CRC-16。当发送多字节数据 (如数据长度和校验和) 时，先发送最低有效字节。

Bootloader 以响应数据包来响应来自主机的每个指令。响应数据包的格式与指令数据包的相似，但响应数据包中使用状态/错误代码，而不是指令代码。可以在第 45 页的 **Table 8** 中查找重要的指令、数据字节及 Bootloader 响应数据包。

10.3 Bootloader 主机的 I2C 数据操作信息

I²C 数据操作是两个器件之间进行的低级别通信。**Figure 43** 和 **Figure 44** 介绍了主机读写 PSoC™的格式。

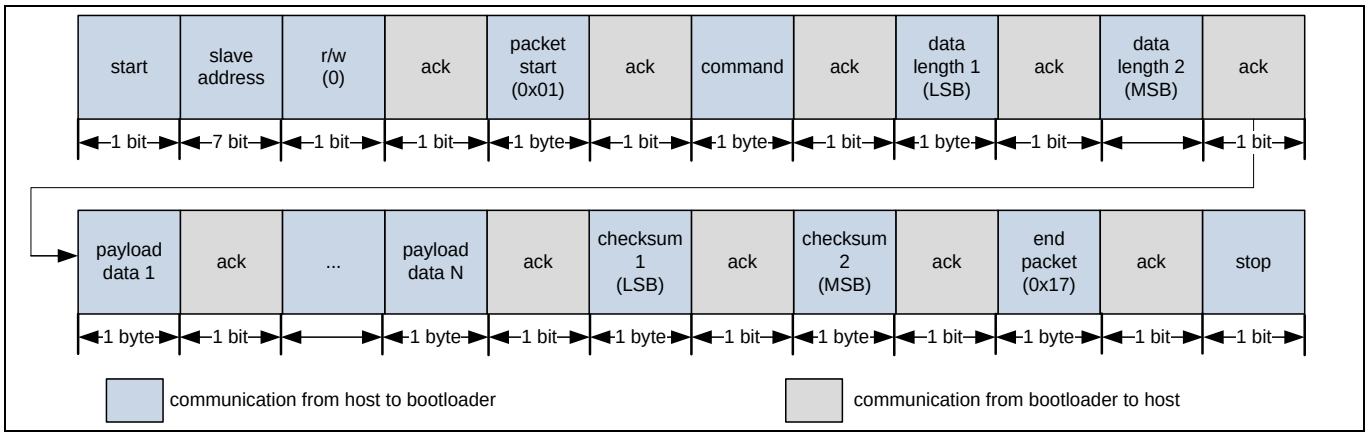


Figure 43 通过 I²C 接口将数据从主机发送到 Bootloader (写入)

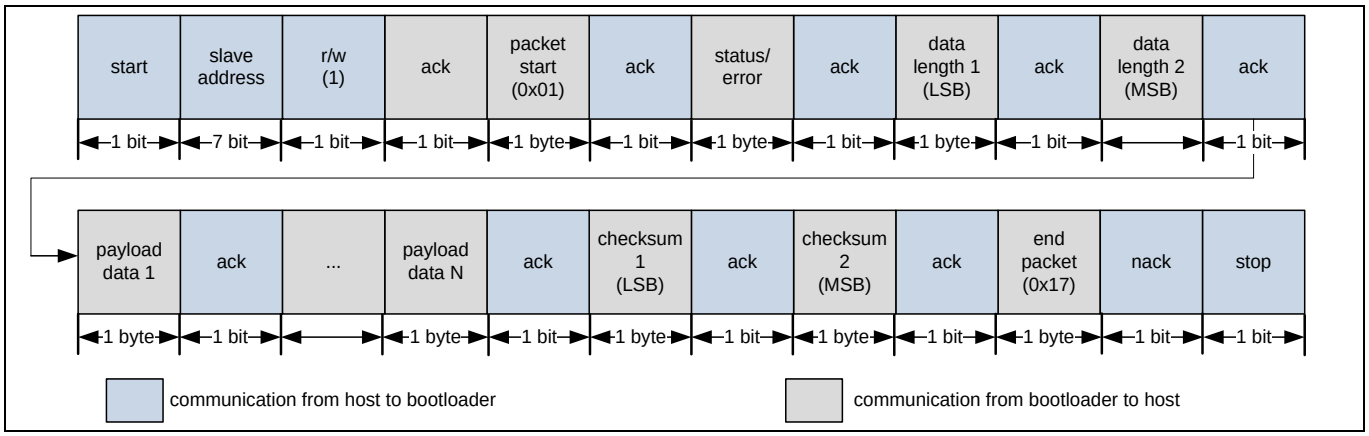


Figure 44 通过 I²C 接口接收来自 Bootloader 的数据 (读取)

10.3.1 指令和状态/错误代码

如前一部分所述，指令数据包和响应数据包具有相同的结构。唯一的区别在于第二个字节包含指令代码还是状态/错误代码。

Table 8 介绍了指令及其预期响应的列表。第 46 页上的 **Table 9** 介绍了状态和错误代码的列表。

Table 8 引导加载指令

指令字节	指令	指令数据包中的数据字节	预期响应的数据字节
0x31	验证校验和	N/A	1 字节：非零或‘0’。 如果是一个非零字节，则应用程序校验和匹配，应用程序有效。 如果是一个零字节，那么校验和不对，应用程序无效。
0x32	获取闪存大小	闪存阵列 ID，一个字节	Bootloadable 闪存的第一行编号，两个字节。 Bootloadable 闪存的最后行编号，两个字节。 这些编号用于所请求的阵列 ID。
0x33	获取应用状态 (仅适用于多应用 Bootloader)	应用编号，一个字节	有效应用编号，一个字节。 活动应用编号，一个字节。 检查指定应用是否有效和是否活动。
0x34	擦除行	闪存阵列 ID，一个字节 闪存行编号，两个字节	N/A。 擦除指定闪存行中的内容。
0x35	同步 Bootloader	N/A	N/A。 将 Bootloader 复位为空白状态。所有缓冲的数据都将被删除。只有 Bootloader 和主机之间不再同步时才需要使用该指令。
0x36	设置活动应用 (仅适用于多应用 Bootloader)	应用编号，一个字节	N/A。 将指定应用设置为活动状态。
0x37	发送数据	需要发送的 N 个数据字节	N/A。 Bootloader 将所接收的数据字节缓存起来，并等待编程行指令。
0x38	进入 Bootloader	N/A	芯片 ID，4 个字节。 芯片版本，一个字节。 Bootloader 版本，3 个字节。 在接收到此指令之前，所有指令都被忽略。
0x39	编程行	闪存阵列 ID，一个字节 闪存行编号，两个字节 需要发送的 N 个数据字节	N/A。 通过使用 Send data (发送数据) 指令将多字节的数据发送到 Bootloader 后，最后数据块会同该指令一起被发送。
0x3A	验证行	闪存阵列 ID，一个字节 闪存行编号，两个字节	行校验和，一个字节。 返回指定行的校验和。
0x3B	退出 Bootloader	N/A	N/A。 此指令没有得到回应。

附录 C：主机/目标通信

Table 9 引导加载状态/ 错误代码 – 指令的可能响应

状态/错误代码	标签	说明
0x00	CYRET_SUCCESS	成功接收并执行指令。
0x02	BOOTLOADER_ERR_VERIFY	闪存验证失败。
0x03	BOOTLOADER_ERR_LENGTH	现有数据量超出了预定范围。
0x04	BOOTLOADER_ERR_DATA	数据形式错误。
0x05	BOOTLOADER_ERR_CMD	无法识别指令。
0x06	BOOTLOADER_ERR_DEVICE	期望器件与检测到的器件不匹配。
0x07	BOOTLOADER_ERR_VERSION	不支持检测到的 Bootloader 版本。
0x08	BOOTLOADER_ERR_CHECKSUM	校验和与期望值不匹配。
0x09	BOOTLOADER_ERR_ARRAY	闪存阵列 ID 无效。
0x0A	BOOTLOADER_ERR_ROW	闪存行编号无效。
0x0C	BOOTLOADER_ERR_APP	应用无效，不能将其设置为活动状态
0x0D	BOOTLOADER_ERR_ACTIVE	当前应用被标记为活动状态。
0x0F	BOOTLOADER_ERR_UNK	发生未知错误。

11 附录 D：主机内核 API

11.1 cybtldr_api2.c / .h

该模块是级别更高的 API，用于处理所有的引导加载操作。其函数可以打开和关闭文件。它为引导加载操作调用 *cybtldr_api.c / .h* API 的函数。当编译基于 GUI 的 Bootloader 主机时，可以使用这些 API。

11.2 cybtldr_parse.c / .h

该模块用于解析*.cyacd* 文件，该文件包含要发送给器件的 Bootloadable 镜像。该模块具有用于设置文件访问、读取文件头、读取行数据和关闭文件的函数。

11.3 cybtldr_api.c / .h

这是一个行级的 API 文件，用于一次性将单行数据发送到 Bootloader 目标。该模块具有用于设置引导加载操作、擦除行、编程行、验证行和结束引导加载操作的功能。**Table 10** 详细描述了该 API 文件的函数。

Table 10 *cybtldr_api.c /.h* 的函数

函数	说明
`CyBtldr_StartBootloadOperation`	使能通信接口并将 Enter Bootloader 指令发送到目标。 通过接收到的响应数据包，可以验证芯片 ID、目标器件的芯片版本以及 Bootloader 版本。
`CyBtldr_ProgramRow`	首先要验证某个行，即将 Get Flash Size (获取闪存大小) 指令发送到目标，以获取目标闪存的特定阵列 ID。响应此指令时，目标将返回该阵列的 Bootloadable 闪存部分起始和结束的行编号。主机读取此响应，并检查指定行是否位于闪存的 Bootloadable 区域内。 如果成功验证该行，主机会将行数据分为适当的大小，并使用 Send Data (发送数据) 指令将它们发送到目标。 Program Row (编程行) 指令随行数据的最后部分发送到目标。
`CyBtldr_VerifyRow`	该函数先验证由特定阵列 ID 和行编号确定的数据行。 如果验证行成功，将为已验证的闪存行发送一个 Verify Row (验证行) 指令。响应该指令时，目标将返回该行的校验和。 根据期望校验和的值验证所返回的校验和。
`CyBtldr_EraseRow`	该函数先验证由特定阵列 ID 和行编号确定的数据行。 如果验证行成功，为已验证的闪存行发送一个 Erase Row (擦除行) 指令。
`CyBtldr_EndBootloadOperation`	发送 Exit Bootload (退出引导加载) 指令，并禁用通信接口。

11.4 cybtldr_command.c / .h

该 API 处理发送到目标的指令数据包结构，并解析来自目标的响应数据包。*cybtldr_api.c / .h* 调用该 API 的函数。例如，为了发送 Enter Bootload (进入引导加载) 指令，`CyBtldr_StartBootloadOperation()` 将调用该 API 的 `CyBtldr_CreateEnterBootloadCmd()`函数。同时它还包含用于发送到目标前计算指令数据包校验和的函数。

12 附录 E：其它主题

12.1 Bootloader 与 HSSP

通过 Bootloader 可以使用通信接口升级用户系统固件。但为了实现完整的闪存升级，包括 Bootloader 闪存区域在内，用户必须使用 SWD 编程器 (源主机串口编程，HSSP)。为了创建 PSoC™ 4 的 HSSP，请参考 **AN84858 – 使用外部微控制器 (HSSP)** 对 PSoC™ 4 进行编程。

12.2 在引导加载过程中，如果发生断电，将导致什么情况?

如果在引导加载过程中发生断电，那么，在下一次复位时，Bootloadable 项目的校验和会与期望值 (存储在闪存最后行内的 Bootloadable 项目校验和) 不再匹配，并且 Bootloadable 项目被视为无效。Bootloader 中继续执行编程，直到成功进行引导加载为止。Bootloader 主机必须发送一个 Start Bootload (启动引导加载) 指令，以重新启动引导加载操作。

12.3 在 Bootloader 和 Bootloadable 项目之间进行跳转，为什么需要复位操作?

PSoC™是一个强大的可配置器件。Bootloader 允许用户修改片上硬件资源及固件。由于其高度可配置的架构，硬件的重配置 (布置、路由、功能) 只在复位状态下执行。因此，Bootloader 需要复位，以实现在 Bootloader 和 Bootloadable 项目之间跳转。

12.4 将应用项目类型从普通 (Normal) 转换为 Bootloadable

如果您已经创建了一个标准 (普通) 项目并想将其转换为 Bootloadable 项目，请将 Bootloadable 组件添加到顶层设计并添加 Bootloader 项目的.*hex* 文件作为依赖属性，如第 12 页上的 **Figure 11** 所示。

如果创建了一个某个普通项目，然后通过将应用类型修改为 **Bootloader** 使该项目变成一个 Bootloader 项目，那么，需要在 *main.c* 中插入 `Bootloader_Start()`函数调用，从而使 Bootloader 项目正常工作。

注意：对于 PSoC™ Creator 3.1，如果您想将一个标准 (普通) 项目转换为 Bootloadable/Bootloader 项目，那么，请将其应用类型修改为 Bootloadable/Bootloader。为实现此操作，请右击 **Project** > **Build setting** > **Code Generation** > **General 选项卡**然后修改 Application Type 项。另外，将 Bootloadable/Bootloader 组件添加到顶层设计。

12.5 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。通过软件控制器件的复位，可以从 Bootloader 跳转到 Bootloadable 项目。该复位也会使调试器接口发生复位，这样，Bootloadable 项目不能在调试器模式下运行。

要想调试某个 Bootloadable 项目，请将它的应用类型修改为“Normal”，然后调试该项目。在完成调试操作后将之转换回 Bootloadable 项目。

另一个选择是在执行 Bootloadable 项目期间，将该 Bootloadable 项目.*hex* 文件编程到器件内，然后使用“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。

12.6 多应用 Bootloader

多应用 Bootloader (MABL) 用来同时将两个 Bootloadable 应用放置在闪存内。可以使用两个相同的应用，以确保器件闪存内始终存在有效的应用。也可以使用两个不同的应用，使得能够通过 Bootloader 指令在该两个应用之间进行切换。不过，此功能有明显的限制，即为每个应用只能占用有效闪存存储器的一半。

附录 E：其它主题

Figure 39 介绍的是闪存存储器中 MABL 的项目布置情况。

按照以下步骤 (它们与执行标准 Bootloader 应用的步骤不同)，可以执行 MABL：

1. 创建一个新的 MABL Bootloader 项目。勾选 Bootloader configuration 窗口中的 Multi-App Bootloader 复选框。

Note: 对于 PSoC™Creator 3.1，将应用类型设置为 Multi-App Bootloader。

2. 将两个 Bootloadable 项目 (Project_A 和 Project_B）添加到工作区内。对于每一个项目，请将相应的依赖关系添加到 MABL 项目内。每个项目都有两个.*cyacd* 文件 — 一个用于闪存的末端，另一个用于闪存的初始端 (请参见 **Figure 39**)：
   - Project_A_1.cyacd 和 Project_A_2.cyacd
   - Project_B_1.cyacd 和 Project_B_2.cyacd
3. 后缀为“1”的.*cyacd* 文件始终占用闪存的前半部分；后缀为“2”的.*cyacd* 文件则占用闪存的后半部分。由于上述原因，只能使用.*cyacd* 文件的特定组合。这些组合包括：
   - Project_A_1.cyacd 和 Project_A_2.cyacd
   - Project_B_1.cyacd 和 Project_B_2.cyacd
   - Project_A_1.cyacd 和 Project_B_2.cyacd
   - Project_B_1.cyacd 和 Project_A_2.cyacd
4. 根据上述某个组合，通过使用 Bootloader 主机应用，按顺序编程多应用 Bootloader 项目的器件，并引导加载各应用 (.*cyacd* 文件)。
5. 请按下述步骤在各应用之间进行切换：
   - 使用 USB 线缆将 PSoC™套件连接至 PC。请确保 Bootloader 处于活动状态。
   - 依次选择 **Start** > **All Programs** >**Cypress** >**Bridge Control Panel**，打开 Bridge Control Panel。选择 I^2C 协议的 KitProg。
   - 通过发送“0x38”指令进入 Bootloader：

     ```
     w 08 01 38 00 00 C7 FF 17

     r 08 x x x x x x x x x x x x x x x x
     ```

   - 要想从 application_1 切换到 application_2，请发送 set_active_application 指令 (0x36)：

     ```
     w 08 01 36 01 00 01 C7 FF 17

     r 08 x x x x x x x
     ```

   - 要想从 application_2 切换到 application_1，请发送 set_active_application 指令 (0x36)：

     ```
     w 08 01 36 01 00 00 C8 FF 17

     r 08 x x x x x x x
     ```

   - 为了切换到应用程序，请发送 exit_bootloader 指令 (0x3B)：

     ```
     w 08 01 3B 00 00 C4 FF 17

     r 08 x x x x x x x
     ```

Table 11 中的实例使用 set_active_application 指令 (从 application_1 到 application_2）来说明这些指令：

附录 E：其它主题

Table 11　指令字节

字节	1	1	1	2	N	2	1
值	08	01	36	01 00	01	C7 FF	17
说明	从设备地址	数据包开始字节	指令	需要发送的字节数量 (最低有效字节优先)	数据字节	校验和 (最低有效字节优先)	数据包结束字节

12.6.1　Bootloader 的存储器要求

在选择 Arm GCC 编译器的“大小”优化下，一个典型的 I²C Bootloader 项目 (包括所有的可选指令在内）大概占用 PSoC™闪存中 4.3 KB 的空间。当编译 Bootloader 项目时，您可以在输出窗口中查看该项目所使用的存储器空间。Bootloadable 项目可以再次使用 Bootloader 项目所使用的 RAM 存储器。

通过取消选择受 Bootloader 组件支持的可选指令，可以稍微减少 Bootloader 项目所占用的存储器容量，如 **Figure 45** 所示。

为了尽可能减少使用闪存存储器，请在*.cydwr* > System 选项卡下将 **Device Configuration Mode** 设置为 **Compressed**，如 **Figure 46** 所示。

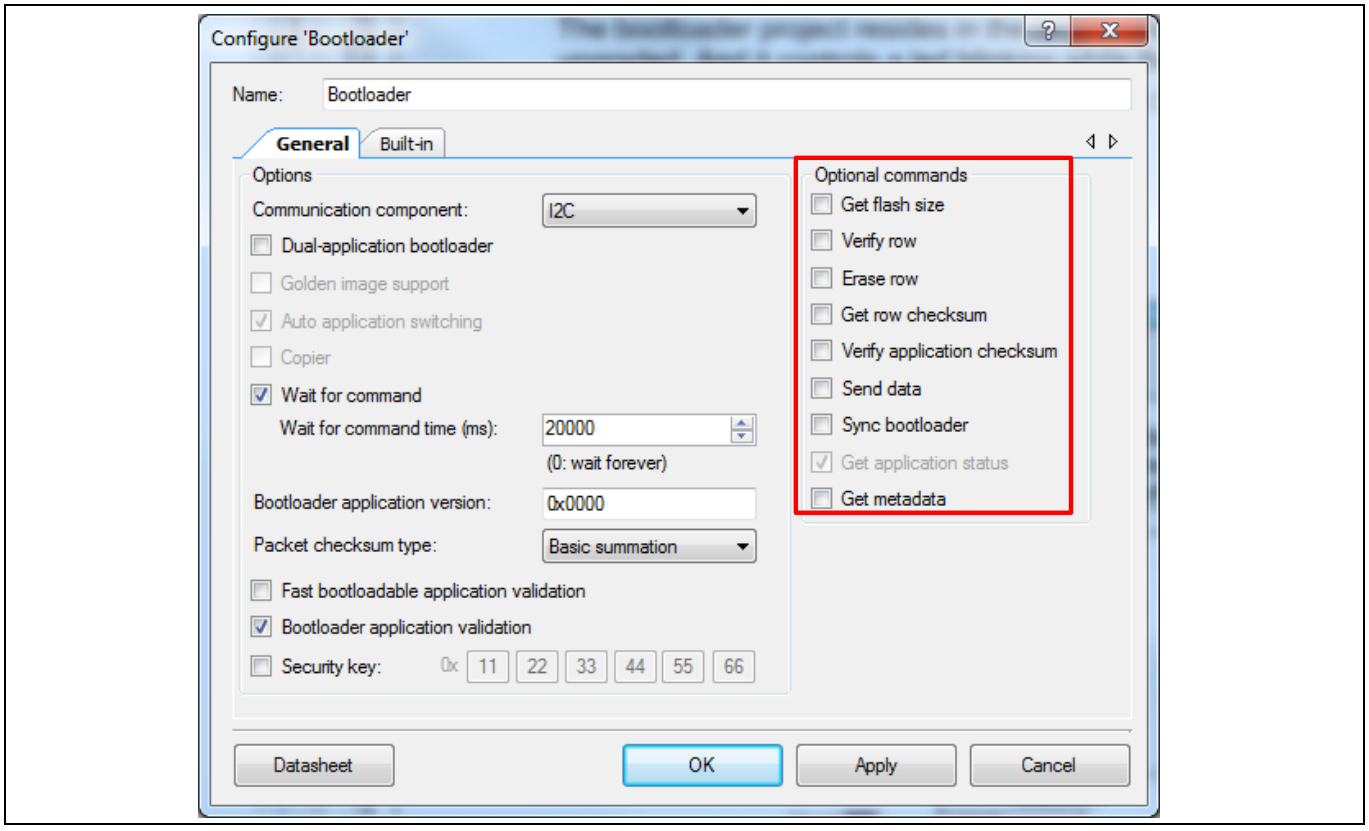


Figure 45　取消选择 Bootloader 组件中的 Optional Commands 项

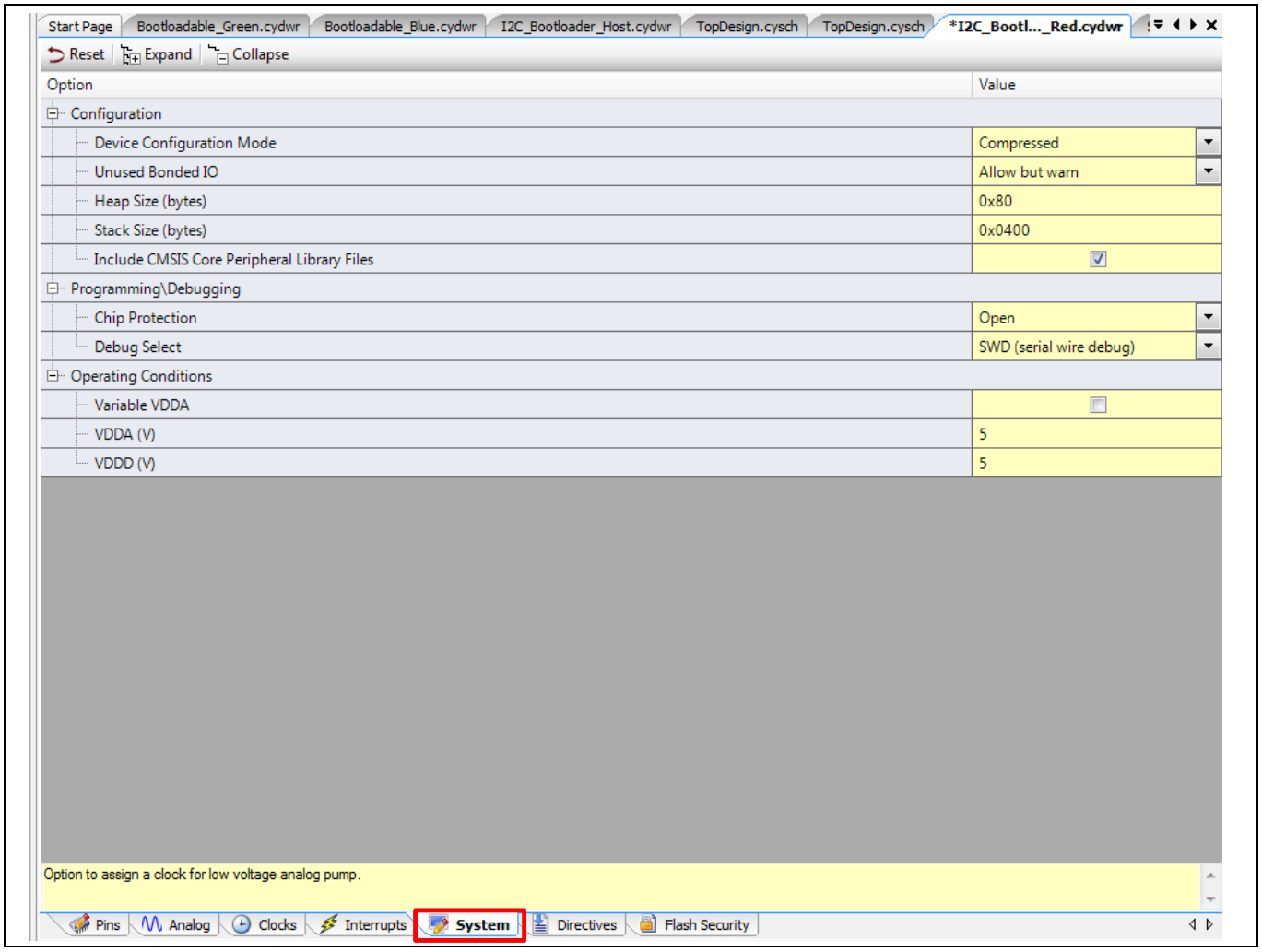


Figure 46　器件配置模式

12.6.2　PSoC™中 PSoC™套件与 MiniProg3 的对比

PSoC™ 4 套件和 **MiniProg3** 都支持 PSoC™的调试/编程和 USB-I²C 桥接器。另外，PSoC™还支持不受 MiniProg3 支持的 USB-UART 桥接器。因此，通过使用 PSoC™套件，可以轻松地进行调试/编程和引导加载操作。更多有关信息，请参考 **CY8CKIT-040 套件指南**和 **CY8CKIT-042 PSoC™ Pioneer 套件指南**。

13 附录 F：套件选择

目标器件	套件名称	用户指南
CY8C42xx	CY8CKIT-042 PSoC™ 4 Pioneer 套件	**CY8CKIT-042**
CY8C40xx-S	CY8CKIT-041-40XX PSoC™ 4 S 系列 Pioneer 套件	**CY8CKIT-041**
CY8C41xx-S	CY8CKIT-041-41XX PSoC™ 4100S CapSense Pioneer 套件	**CY8CKIT-041**
CY8C40xx	CY8CKIT-040 Pioneer 套件	**CY8CKIT-040**
CY8C41xx-PS	CY8CKIT-147 PSoC™ 4100PS Prototyping 套件	**CY8CKIT-147**
CY8C4100S Plus	CY8CKIT-149 PSoC™ 4100S Plus Prototyping 套件	**CY8CKIT-149**

文档修订记录

版本	提交日期	变更说明
**	2013-06-13	新建应用手册。
*A	2014-04-21	本文档版本号为 Rev*A，译自英文版 001-86526 Rev*A。
*B	2015-04-15	本文档版本号为 Rev*B，译自英文版 001-86526 Rev*C。
*C	2016-05-10	本文档版本号为 Rev*C，译自英文版 001-86526 Rev*D。
*D	2017-02-09	本文档版本号为 Rev *D， 修改 Rev *C 第一节中相关应用笔记的超级链接错误。 版权日期修改为 2013-2017， 原来为 2013-2016.
*E	2017-07-19	更新徽标和版权。
*F	2019-03-20	本文档版本号为 Rev. *F，译自英文版 001-86526 Rev. *G。 更新了页眉 更新了 PSoC™ 4100S Plus 器件 更新了 PSoC™ Creator 4.2 的图示和信息 在**附录 A** 中添加了 PSoC™ 4100S Plus 的信息 将 ARM_GCC_493 换成了 ARM_GCC_541 更新了 **Figure 4**，**Figure 11**，**Figure 11** 和 **Figure 21** 在**附录 F** 中添加了 PSoC™ 4100S Plus 更新了 PSoC™ 4100PS 并且移除了 PSoC 模拟协处理器 更新了 PSoC™ Creator 4.2 的工程
*G	2022-11-16	翻译自: 001-86526 Rev. *H

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

Edition 2022-11-16
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
001-89408 Rev. *G

重要提示

本应用说明中包含的信息仅作为产品实施的提示， 在任何情况下均不应视为对产品特定功能、产品状 况或产品质量的描述或保证。在产品实施之前，本 应用说明的接收方必须在实际应用中验证此处给出 的所有功能和其他技术信息。对于本应用说明中提 供的所有信息，英飞凌科技公司不承担任何形式的 任何保证和责任（包括但不限于不侵犯任何第三方 知识产权的保证）。

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。

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当被用于任何一项一旦产品失效或者产品使用的后领域。