使用外部微控制器对 PSoC™ 4 进行编程 (HSSP)

关于本文档

范围及目的

AN84858 向您介绍了如何通过模块化的 C 代码使用外部微控制器对 PSoC™ 4 器件进行编程。在该过程中，又称为主机发起串行编程 (HSSP)，主机微控制器通过串行线调试 (SWD) 接口对 PSoC™ 4 进行编程。本应用笔记使用 C 代码进行编程，使其移植到其它微控制器的代价尽量小，从而为 PSoC™ 4 的 HSSP 应用开发加快速度。

更多代码示例？我们明白。

如需寻找包含上百 PSoC™代码示例并有不断更新的网上资源，请浏览我们的**代码示例网页**。您还可以在**此处**观看 PSoC™视频库。

目录

关于本文档 **1**
目录 **1**
1 简介 **3**
1.1 编程器的类型 3
1.2 术语和定义 3
2 HSSP 固件架构 **5**
2.1 SWD 协议物理层 6
2.2 SWD 协议数据包层 6
2.3 提取编程数据 6
2.4 HSSP 编程步骤 7
2.5 HSSP 超时参数 7
2.6 HSSP 编程数据 8
2.7 主要应用代码 8
2.8 HSSP 错误状态 8
3 用于 HSSP 编程的硬件连接 **10**
4 将 HSSP 应用移植到主机编程器 **11**
4.1 需要移植的文件 11
4.2 移植时要求的代码更改 11
5 计算 HSSP 超时参数 **13**
5.1 DEVICE_ACQUIRE_TIMEOUT 13
5.2 SROM_POLLING_TIMEOUT 14
5.3 XRES_PULSE_100US 15
6 接收 HSSP 编程数据的接口 **17**
7 HSSP 时序验证 **19**
8 电源循环模式编程 **20**
9 测试示例项目 **21**
9.1 对于 CY8CKIT-038 PSoC™ 4 开发套件 21

目录

9.2 对于具有板上 PSoC™ 5LP 编程器 (KitProg) 的套件 21
9.3 对于具有板上 PSoC™ 5LP 编程器的套件 (KitProg2) 25
10 解决有关 HSSP 问题的提示和技巧 29
11 总结 30
12 相关文档 31
12.1 应用笔记 31
12.2 编程规范 31
12.3 架构技术参考手册 31
12.4 网页 31
13 随附的项目 32
14 附录 A. Hex 文件解析器应用 33
14.1 使用 Hex 文件解析应用 33
14.2 将所生成的文件添加到 PSoC™ Creator 示例项目中 34
14.2.1 对于 PSoC™ 4xxxL/ PSoC™ 41x8_BLE/ 42x8_BLE/ 41XXS Plus 器件系列，请使用 *HexImage.c*、*HexImage.h* 文件: 35
15 附录 B. SROM 请求的状态代码 37
16 附录 C. HSSP 错误状态寄存器的位字段定义 38
16.1 位[2:0] — SWD 确认响应 (SWD ACK [2:0]) 38
16.2 位 3 — SWD 读取数据奇偶校验错误 38
16.3 位 4 — 端口获取超时 38
16.4 位 5 — SROM 轮询超时错误 38
16.5 位 6 — 验证失败 39
16.6 位 7 — 跃变错误 39
17 附录 D. HSSP 函数 40
文档修订记录 42

1 简介

PSoC™ 4 器件编程特指使用外部主机编程器对 PSoC™ 4 中的非易失性存储器进行的编程。主机编程器可以是由英飞凌提供的 (**CY8CKIT-002 MiniProg3**)、第三方编程器或一个自定义的编程器 (例如，板上的微控制器)。本应用笔记说明了如何使用主机编程器对 PSoC™ 4 器件进行编程。有关 PSoC™ 4 架构和使用 PSoC™ Creator 软件创建 PSoC™ 4 的项目的详细信息，请参阅 **AN79953 — PSoC™ 4 入门。**

1.1 编程器的类型

您所选的编程器类型取决于产品开发过程的具体阶段:

原型设计: 编程器必须能够执行以下两个功能:

1. 编程器件。
2. 调试器件，排除应用中的故障。

为了完成编程和调试操作，原型设计阶段中所用的编成器必须能与集成开发环境 (IDE) (例如 **PSoC™ Creator**) 交互。例如，**英飞凌的 MiniProg3** 或 **PSoC™ 5LP 原型套件**可以作为低成本编程器/调试器。

生产: 需要一个能够编程多个器件的编程器。该编程器通过解析 Hex 文件提取必要的信息，并且通过编程接口 (如 SWD) 进行编程。

编程器可以分为以下两大类:

- 在线编程器可直接对安装在终端应用 PCB 上的目标器件进行编程。您可以将外部编程器连接到器件的编程引脚，从而实现在线编程。
- 插座编程器要求将目标器件安装在编程器硬件的插座上，以进行编程操作。编程完成后，将目标器件焊接到终端应用的 PCB 上。几乎所有的**第三方生产编程器**都是插座类型。

无论是在线编程还是插座编程中，编程器都要实现 HSSP 算法并生成信号，从而用 Hex 文件中的数据进行编程。

本应用笔记提供了 C 代码，用于实现 HSSP 编程器。您只需要很小的改变就可以将 C 代码移植到任何主机微控制器内。通过移植操作， 您可以缩短开发 PSoC™ 4 器件的 HSSP 应用所需要的时间。本应用手册所提供的项目使用 PSoC™ 5LP 器件作为主机编程器，对目标器件进行编程。

参阅本应用笔记之前，请您先查看**相关文档**一节中所列出的相应器件编程规范文档。本文档说明了对 PSoC™ 4 器件进行编程时所需要的编程接口、编程算法、硬件连接和电气时序等规范。本应用笔记是编程规范的实际应用。

1.2 术语和定义

- **串行线调试 (SWD):** 由 ARM 开发的 SWD 协议仅使用两条线路 (即 SWDCLK (时钟) 和 SWDIO (双向数据线)) 来进行编程和调试。
- **调试访问端口 (DAP):** DAP 是 SWD 和 PSoC™ 4 中 CPU 之间的编程/调试接口。它包含一个调试端口 (DP) 和一个访问端口 (AP)。
  - DP 为编程器/调试器提供了物理连接。
  - AP 提供了 DAP 模块与 Cortex-M0 CPU、闪存等之间的接口。
- **HSSP:** HSSP 是指使用主机微控制器对板上目标器件进行编程。通过 SWD 接口对目标器件进行编程。在本应用手册中，HSSP 使用了 Bit-banging 对目标器件进行编程。Bit-banging 编程是指使用保存在主机编程器中的软件代码控制编程引脚的技术。
- **Bootloading 和 HSSP 之间的差异**: 在嵌入式系统中，Bootloader 也经常使用于更新系统固件。Bootloading 和 HSSP 的不同之处主要在以下几个方面:

简介

- Bootloader 通过标准的通信协议对器件的闪存进行更新。Bootloader 仅能更新闪存内特定的 Bootloadable 区域。
- HSSP 则支持对目标器件的整个闪存进行编程。
- Bootloader 可以使用任一标准的通信接口 (如 USB、I^2C、SPI 和 UART) 更新固件，而 HSSP 使用 SWD 或 JTAG 接口对闪存进行编程。PSoC™ 4 仅支持 SWD 接口。

2 HSSP 固件架构

目标器件的 HSSP 用 C 代码分为多个层次来实现。这些层次包括:

1. SWD 协议物理层
2. SWD 协议数据包层
3. HSSP 编程步骤层

请参见 **Figure 1**，了解这些层次之间的控制流程。

Note: *欲了解有关实现该固件的信息，请参考本应用笔记附带的 A_Hssp_Programmer 项目，该项目将 PSoC™ 5LP 作为外部主机编程器使用。*

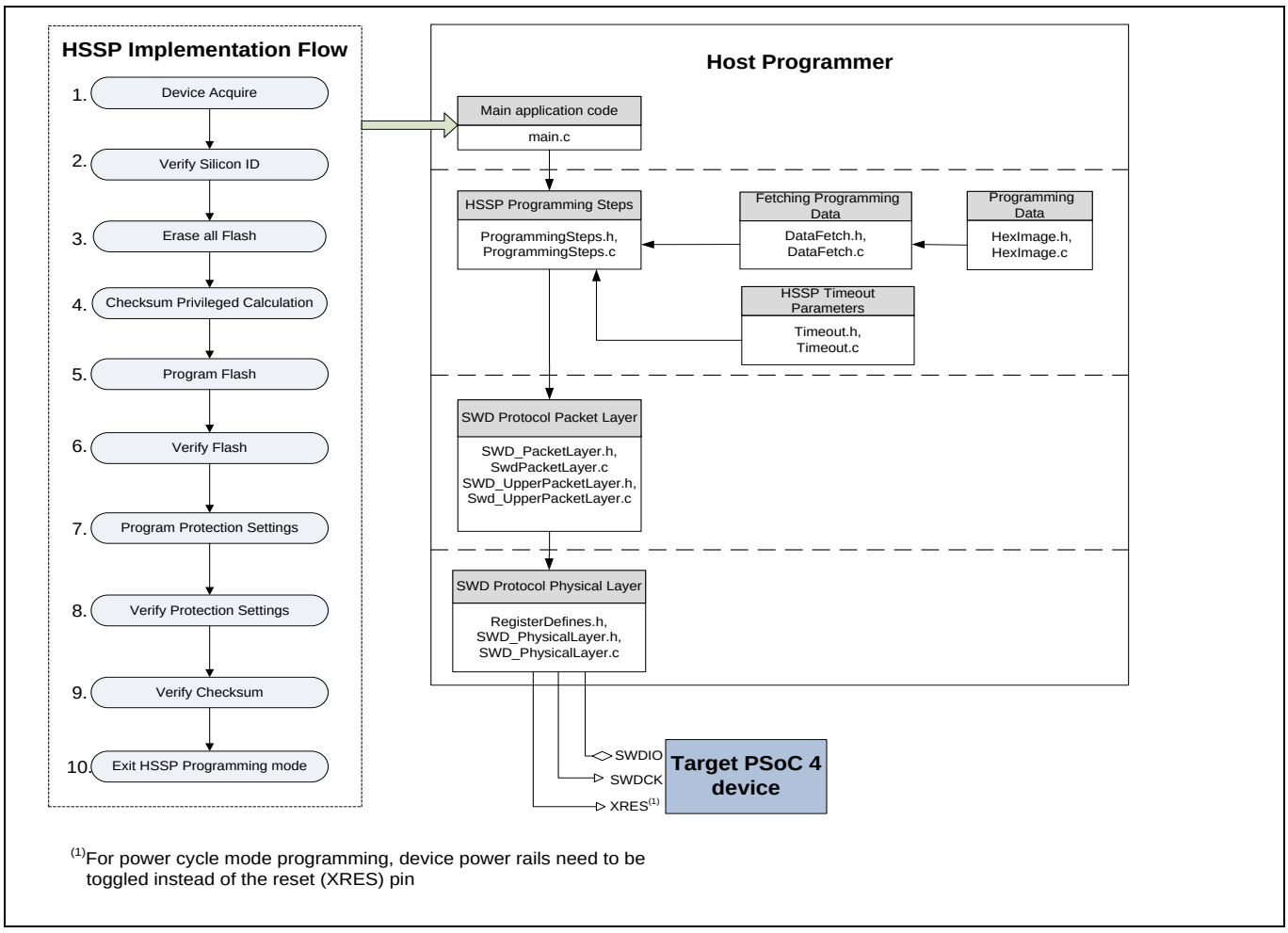


Figure 1 HSSP 固件架构

HSSP 编程步骤层使用“提取编程数据”接口从数据源 (例如: 由 I^2C、SPI、UART 或 USB 等任何通信接口或主机闪存 (如 **Figure 1** 所示) 提供的 Hex 文件数据) 提取编程数据。此外，“HSSP 编程步骤”层使用 HSSP 超时参数接口配置在编程 API 过程中的超时。

以下各节对固件架构所用的所有层次 (如 **Figure 1** 所示) 提供了说明。

2.1 SWD 协议物理层

Table 1 描述了组成 SWD 协议物理层的文件。

Table 1 组成 SWD 协议物理层的文件

源文件	说明
RegisterDefines (*.h* 文件)	该文件定义了编程引脚的端口号、引脚号、输入/输出寄存器和驱动模式寄存器。
SWD_PhysicalLayer (*.c* 和 *.h* 文件)	这些文件包含用于操作编程引脚的宏和函数。在 *RegisterDefines.h* 文件中定义了这些引脚。

上述文件中的代码适用于 PSoC™ 5LP 主机微控制器。如果将这些文件移植到任何其它的主机微控制器内，那么应当对所有函数和宏进行适当的修改。

*Note: 有关主机端上引脚配置的详细信息，请在**相关文档**所列出的相应器件编程规范文档中查看“引脚名称和要求”部分。*

2.2 SWD 协议数据包层

Table 2 介绍了组成 SWD 协议数据包层的文件。

Table 2 组成 SWD 协议数据包层的文件

源文件	说明
SWD_PacketLayer (*.c* 和 *.h* 文件)	这些文件定义了根据 SWD 协议传送 SWD 读取和 SWD 写入数据包的数据包子程序。
SWD_UpperPacketLayer (*.c* 和 *.h* 文件)	这些文件使用 SWD_PacketLayer 中所定义的函数直接读取和写入 DAP 寄存器和 CPU 地址空间。

ProgrammingSteps.c 文件中的函数直接调用本层所定义的各函数。

这些 SWD 数据包函数使用三个全局变量 (即 `swd_PacketHeader`、`swd_PacketAck` 和 `swd_PacketData[]`) 实现 。这三个全局变量可由顶层文件中的函数访问，如 Figure 1 所示。

2.3 提取编程数据

Table 3 描述了提取编程数据以供给上层函数的文件。

Table 3 提取数据层文件

源文件	说明
DataFetch (*.c* 和 *.h* 文件)	这些文件包含了各个子程序，用于提取 *HexImage.c* 文件中的编程数据，然后将该数据传送给 *ProgrammingSteps.c* 文件中的函数。 编程数据包括: 闪存行数据、闪存保护数据、芯片保护数据、芯片 ID、 校验和与闪存行总数。

请根据获取 HSSP 编程数据的方法修改函数的定义。

*Note: 请参阅“**接收 HSSP 编程数据的接口**”一节，了解如何修改 HSSP 源代码。*

2.4 HSSP 编程步骤

该层包括 *ProgrammingSteps.c* 和 *ProgrammingSteps.h* 文件。这两个文件包含 HSSP 应用的顶层函数。下面各步骤介绍了这些函数:

1. **获取器件:** 在该步骤中，器件复位后，通过 SWD 接口发送一个特定的序列来获取器件。因此，主机编程器可以控制 Cortex-M0 CPU 以及其它系统资源 (如: SRAM、寄存器)。对闪存进行擦除/写入操作之前，PSoC™ 40xx、PSoC™ 4xx7_BLE、PSoC™ 40xxS、PSoC™ 41xxS 和 PSoC™ 4100PS 等器件系列需要将内部主振荡器 (IMO) 频率设置为 48 MHz。PSoC™ 40xx 和 PSoC™ 4xx7_BLE 系列的获取器件程序也包含此操作。

Note: *PSoC™ 4000 系列中的某些器件不使用专用的复位 (XRES) 引脚，因此，需要通过切换器件的电源来实现复位。这种编程被称为电源循环模式编程。有关修改代码以进行电源循环模式编程的详细信息，请参考**电源循环模式编程**一节。*

2. **验证芯片 ID:** 该步骤验证所获取的器件 ID 是否与 Hex 文件中的信息相同。
3. **擦除所有闪存:** 擦除所有用户行以及相应的闪存保护数据。
4. **校验和特权计算:** 擦除所有用户行后，该步骤将计算特权行的校验和。该校验和用于验证**步骤 9** 中用户行的校验和。
5. **编程闪存:** 该步骤通过使用 Hex 文件中的编程数据并调用 SROM API 对闪存进行编程。
6. **验证闪存**: 该步骤验证 Hex 文件中的数据是否与上一步所编程的闪存数据相匹配。该步骤是可选的，但强烈推荐您实现。
7. **编程保护设置:** 该步骤将 Hex 文件中的行保护设置和芯片保护设置写入到特定的闪存区域中。
8. **验证保护设置:** 验证这两个保护设置是否与 Hex 文件中的设置相匹配。
9. **验证校验和**: 该步骤验证闪存内的用户数据校验和是否与 Hex 文件中的校验和相匹配。这里使用了步骤 4 中计算出的特权行校验和。
10. **退出 HSSP 编程模式:** 此步骤使目标目标器件退出编程模式。

每个步骤由从各个基本 SWD 指令组成的函数描述。有关详细内容，请查看**相关文档**一节中所列出的相应器件编程规范文档。

ProgrammingSteps.h 文件所声明的函数可以访问 SWD 协议数据包层、数据提取层及超时层的函数、定义和全局变量。*ProgrammingSteps.c* 和 *ProgrammingSteps.h* 等文件中的函数包含了编程目标器件时所需的所有步骤。

2.5 HSSP 超时参数

Table 4 组成超时层的文件

源文件	说明
Timeout (*.c* 和 *.h* 文件)	这两个文件包含了用于 HSSP 的时间纪录 (timestamp) 定义和延迟程序。

时间纪录的定义来自“器件编程规范”中所提供的电气时序规范。这些时间纪录参数定义在 *Timeout.h* 文件中列出的值适用于在 63 MHz 时钟频率下工作的 PSoC™ 5LP 主机编程器。

时间纪录定义和延迟程序用于 *ProgrammingSteps.c* 文件中的函数定义。

请参见**计算 HSSP 超时参数**一节，了解如何为特定的主机编程器计算时间纪录参数。

2.6 HSSP 编程数据

Table 5 介绍了含有存储在主机编程器中的编程数据的文件。

Table 5 包含编程数据的文件

源文件	说明
HexImage (*.c* 和 *.h* 文件)	这些文件包含编程到目标器件中的数据。 这些文件还包含用于 HSSP 编程的目标器件参数，如: 芯片 ID、校验和、闪存行总数以及每一闪存行中的字节数量。

这些文件中的数据被存储在 PSoC™ 5LP 闪存内，作为一个常量阵列使用。

这些文件由本应用笔记所提供的 C#应用 Hex 文件解析应用生成。该应用程序通过将目标 hex 文件作为输入生成这些文件。**附录 A** 提供了有关该应用程序的详细信息。

有关 Hex 文件格式的详细内容，请在**相关文档**一节所列出的相应器件编程规范文档中查看“附录 B. Intel Hex 文件格式”部分。

对于缺少存储容量 (用于在片上存储器中存储编程数据) 的主机编程器，则不需要 HexImage.c 和 *HexImage.h* 文件。在这种情况下，HSSP 编程数据通常以数据包为单位通过一个通信接口 (如: I^2C、UART 或 USB) 发送到主机上。

*Note: 欲了解有关通过所用方法修改 HSSP 源代码来取得编程数据的信息，请参见**接收 HSSP 编程数据的接口**一节。*

2.7 主要应用代码

main.c 文件是主要的应用代码。它按照 **Figure 1** 所示的顺序调用顶层 HSSP 编程步骤。*main.c* 文件中的 `ProgramDevice()`函数执行所有步骤。您必须成功执行前一步操作才能实现下一步操作。

如果任何步骤执行失败 (FAILURE)，则 HSSP 操作将被中止。此时，可使用 `ReadHsspErrorStatus()`函数返回的错误状态确定发生错误的原因。附加的 HSSP 项目中的字符 LCD 显示了 HSSP 操作和错误状态寄存器的状态。

Note: 字符 LCD 和 `Pin_Start` 程序特定于 PSoC™ 5LP 主机编程器。因此，应当根据其他主机编程器的要求修改它们。

2.8 HSSP 错误状态

HSSP 应用中的任何顶层步骤返回失败状态时，将从主要应用代码调用 `ReadHsspErrorStatus()`函数，以获取错误的详细信息。

此函数将返回状态字节。根据位字段的定义，可以从状态字节推断错误的信息。请查看 **Figure 2** 了解此函数所返回的位字段。

HSSP 固件架构

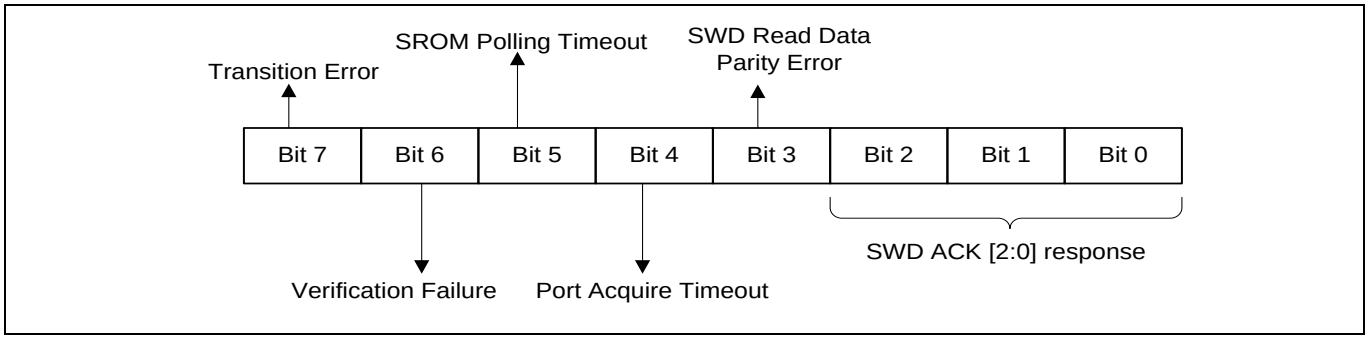


Figure 2 HSSP 错误状态寄存器

Table 6 介绍了该状态寄存器的位字段定义。

Table 6 HSSP 错误状态寄存器

位	字段名称	说明
[2:0]	SWD ACK	这三位存储先前 SWD 数据操作的确认响应。
3	SWD 读取数据奇偶校验错误	如果该位置位，则表示主机收到的数据具有奇偶错误。
4	端口获取超时	如果该位置位，则表示在超时期间没有获得器件。
5	SROM 轮询超时	如果该位被置位，则表示 SROM 操作超过了一秒。
6	验证失败	在多个步骤中都会置位该位。根据失败发生的步骤可以推断错误原因。
7.	跃变错误	如果该位置位，则表示从芯片和 Hex 文件中读取的芯片保护设置指示了一个错误的跃变。

欲了解有关该寄存器的更多信息以及位字段的解释，请查看**附录 C**. HSSP 错误状态寄存器的位字段定义部分。

3 用于 HSSP 编程的硬件连接

可使用 SWD 接口引脚 (SWDIO、SWDCK) 和外部复位引脚 (XRES) 对 PSoC™ 4 中的 HSSP 进行编程。编程规范文档 (本文档的**相关文档**章节中所列出) 中标题为“物理层”的部分详细介绍了主机编程器引脚驱动模式的要求。**Figure 3** 显示了主机编程器和目标 4 器件之间所需要的基本硬件连接。

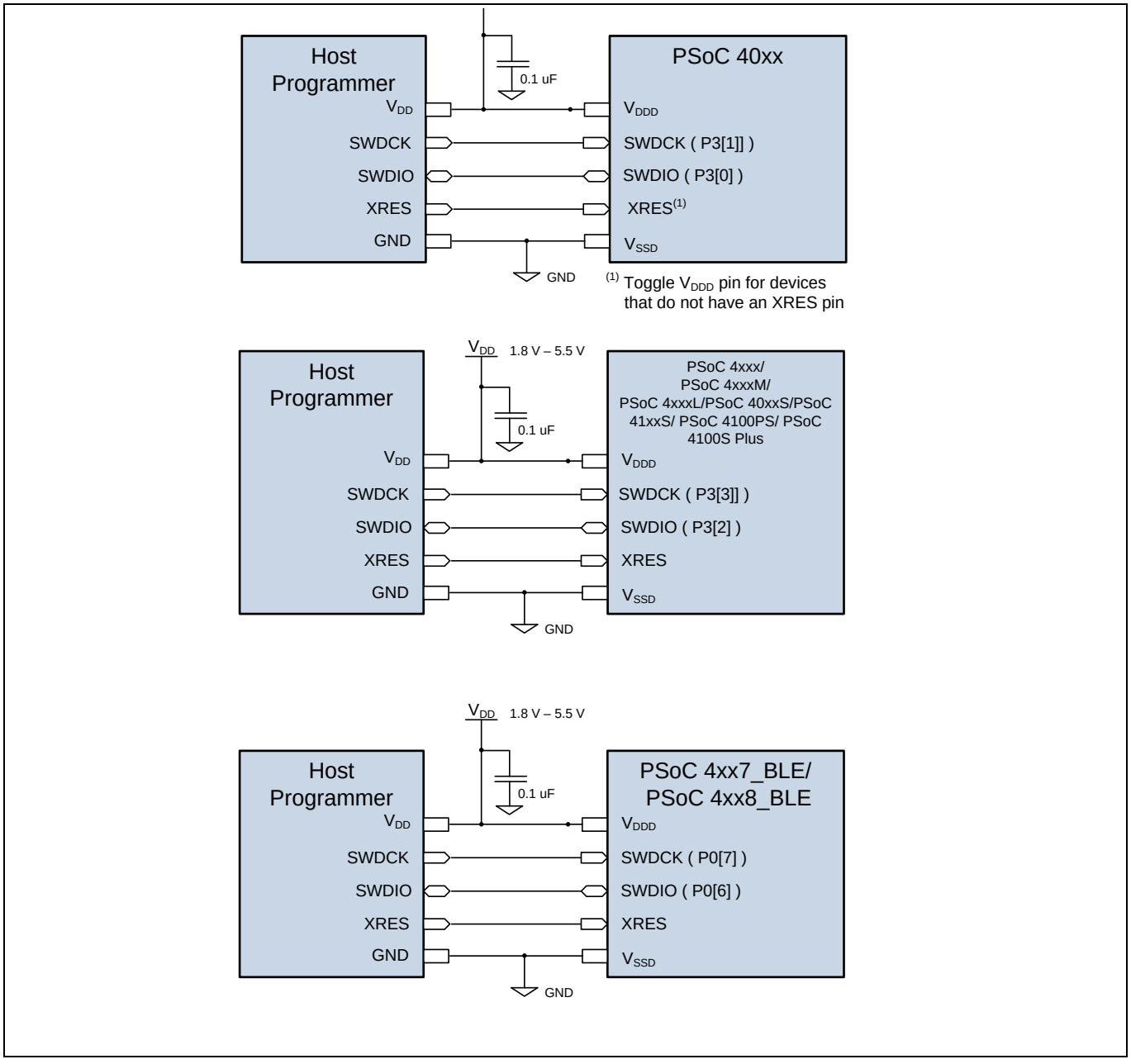


Figure 3 主机编程器和目标器件之间的基本连接

4 将 HSSP 应用移植到主机编程器

本应用笔记所提供的 A_Hssp_Programmer 项目将 PSoC™ 5LP 作为主机编程器使用。在 HSSP 应用中，主机编程器可以是任何微控制器。本节说明了将 HSSP 应用代码移植到特定主机以编程目标器件时所需要的更改。

4.1 需要移植的文件

基于 PSoC™ 5LP 主机编程器的项目包含特定于 PSoC™ 5LP 的文件。当将 HSSP 应用代码移植到其它任何主机编程器时，您只需移植 **Table 7** 中列出的文件。

Table 7 需要移植的文件

需要移植的头文件	需要移植的源文件
SWD_PhysicalLayer.h	*SWD_PhysicalLayer.c*
SWD_PacketLayer.h	*SWD_PacketLayer.c*
SWD_UpperPacketLayer.h	*SWD_UpperPacketLayer.c*
Timeout.h	*Timeout.c*
HexImage.h	*HexImage.c*
DataFetch.h	*DataFetch.c*
ProgrammingSteps.h	*ProgrammingSteps.c*
RegisterDefines.h	

4.2 移植时要求的代码更改

将附带的 HSSP 应用代码移植到任何主机编程器 (PSoC™ 5LP 除外) 时，请对每个文件进行以下更改。请注意，进行移植时，仅要修改下面各文件:

- ***RegisterDefines.h***: 根据所使用的主机编程器，需要修改头文件中的端口号、引脚号、掩码值、输出寄存器、输入寄存器和驱动模式寄存器等定义。
- ***SWD_PhysicalLayer.h***: *SWD_PhysicalLayer.c* 文件中的函数定义使用了该头文件所定义的所有 Bit-banging 宏。根据所使用的主机编程器修改这些宏。
- ***SWD_PhysicalLayer.c***: 根据所使用的主机编程器对该文件所定义的所有 Bit-banging 函数进行更改。
- ***Timeout.h:*** 根据所使用的主机编程器修改 **Table 8** 中介绍的三个超时参数的定义。

Table 8 时序参数

序号	时序参数
1	XRES_PULSE_100US
2	DEVICE_ACQUIRE_TIMEOUT
3	SROM_POLLING_TIMEOUT

欲了解如何为特定的主机编程器计算这些超时参数，请参见**计算 HSSP 超时参数**一节。

- ***HexImage.c*、*HexImage.h***: 这些文件包含编程到目标器件内的常量阵列数据。如果使用 PSoC™ 5LP 主机编程器，编程数据将存储在主机 PSoC™ 5LP 的闪存中。

某些主机编程器可能缺少用于在片上存储器中存储编程数据的存储空间。但它们可以使用通信接口 (如 USB、SPI 或 UART) 获取编程数据。在这种情况下，请移除这些文件。

- ***DataFetch.c***: 应该根据获取编程数据的方法进行修改函数的定义。

将 HSSP 应用移植到主机编程器

欲了解如何根据提取编程数据的方法修改 HSSP 源代码，请参见**接收 HSSP 编程数据的接口**一节。

- ***main.c***: 用于字符 LCD 的 API 和 *main.c* 内的 Pin_Start 引脚专门用于 PSoC™ 5LP 主机编程器。因此，当移植到任何其他主机编程器时，应该移除或修改它们。

在上述所有指定文件中，需要根据主机编程器修改的代码前面是以下各注释。这样有助于您确定需要修改的代码。

```
/******USER ATTENTION REQUIRED******/

/******HOST PROCESSOR SPECIFIC******/
```

5 计算 HSSP 超时参数

根据所使用的主机编程器修改 *Timeout.h* 文件中定义的超时参数的值。

本应用笔记提供了一个名为“C_Hssp_TimeoutCalc”的单独测试项目。该项目说明了如何计算 PSoC™ 5LP 主机编程器的超时参数。可为任何其他主机编程器创建一个类似的测试项目，以计算这些超时值。

该测试项目在两个文件中提供了测试函数:*TimeoutCalc.h* 和 *TimeoutCalc.c*。这些测试函数在执行代码过程中切换一个测试引脚。通过使用示波器测量测试引脚上信号的低脉冲宽度，可以测量出超时参数值。

要想计算超时参数值，请参见项目的 *TimeoutCalc.h* 头文件中宏定义的详细说明。以下内容论述了每个超时参数值的重要性:

5.1 DEVICE_ACQUIRE_TIMEOUT

该宏用于 *ProgrammingSteps.c* 文件中的 `DeviceAcquire()`函数。为了编程器件，在使用 XRES 引脚进行器件复位后，必须在获取器件的最长许可时间内获取该器件。**Table 9** 显示了获取器件所需的最长许可时间。

Table 9 获取器件超时

器件系列	超时 (ms) [1]
PSoC™ 40xx	2.0
PSoC™ 41/42xx	1.5
PSoC™ 4xxxM	2.0
PSoC™ 4xxxL	2.0
PSoC™ 4xx7_BLE/PSoC™ 4xx8_BLE	2.0
PSoC™ 4000S/ PSoC™ 4100S/ PSoC™ 4100PS	2.0
PSoC™ 4100S Plus	5.0

*Note: 建议 SWDCK 时钟的最低频率为 1.5 MHz。此频率可满足时序的要求，从而可以获取器件。有关详细内容，请参见**相关文档**一节所列出的相应器件编程规范文档中“步骤 1. 获取芯片”部分。*

器件获取的序列包含以下两个步骤:

1. 进行线路复位 (使用标准的 ARM 指令复位调试访问端口 (DAP))。
2. 读取 DAP。

DEVICE_ACQUIRE_TIMEOUT 表示器件复位后的特定时间内 (如 **Table 9** 所示)，主机可发送器件获取序列的最大次数。

要想计算此宏，请取消项目中 *main.c* 文件的 `TestModeTimeout()`函数，然后进行编程器件。

发送器件获取序列的一个迭代后，使用示波器测量测试引脚的低脉冲宽度。然后，使用以下公式计算宏的值:

DEVICE_ACQUIRE_TIMEOUT = (X ms/低脉冲宽度)，其中 X 是获取器件的最长许可时间，如 **Table 9** 所示。

1 如要进行电源循环编程，则需要更长的获取器件超时，这个时间为 30 ms。

计算 HSSP 超时参数

Code Listing 1　　计算 DEVICE_ACQUIRE_TIMEOUT 参数

```
Unsignedshort timestamp = 0;
Unsignedlongchip_DAP_Id = 0;

/* Make the pin LOW before sending SWD clock train */
TESTPIN_OUTPUT_LOW;

for(timestamp = 0; timestamp < 1; timestamp++)
{
       Swd_LineReset();
       Read_DAP(DPACC_DP_IDCODE_READ, &chip_DAP_Id);
}
/* Make the pin HIGH after sending SWD clock train */
TESTPIN_OUTPUT_HIGH;
```

5.2　　SROM_POLLING_TIMEOUT

在 HSSP 编程过程中，该宏用于 SROM 轮询操作。通过目标器件的 SROM 请求，可以使用该宏轮询非易失性存储器的读取和写入操作的结果。

当主机通过 SWD 接口请求一个 SROM 系统调用时，它在读取 CPUSS_SYSREQ 寄存器状态时最多等待一秒的时间。如果未得到响应，主机将中止 HSSP 操作。

SROM_POLLING_TIMEOUT 表示在 1 秒的时间内可读取 CPUSS_SYSREQ 寄存器的最大次数。

要想计算此宏，取消项目中 main.c 文件的 `TestSromPollingTimeout()`函数，并进行编程器件。

发送 SROM 轮询序列的 10 个迭代后，使用示波器测量测试引脚的低脉冲宽度。然后，使用以下公式计算宏的值:

SROM_POLLING_TIMEOUT = (1 s/低脉冲宽度) *10

Code Listing 2　　计算 SROM_POLLING_TIMEOUT 参数

```
Unsignedshort timestamp = 0;
UnsignedlongstatusCode = 0;

/* Make the pin low before sending SWD clock train */
TESTPIN_OUTPUT_LOW;

for(timestamp = 0; timestamp < 10; timestamp++)
{
```

计算 HSSP 超时参数

Code Listing 2　计算 SROM_POLLING_TIMEOUT 参数

```
      Read_IO (CPUSS_SYSREQ, &statusCode);


/*performing SROM_SYSREQ_BIT | SROM_PRIVILEGED_BIT */
statusCode &= (SROM_SYSREQ_BIT | SROM_PRIVILEGED_BIT);
}
/* Make the pin high after sending SWD clock train */
TESTPIN_OUTPUT_HIGH;
```

此宏表示在将 SROM_SYSREQ_BIT 和 SROM_PRIVILEGED_BIT 都设置为 0 时，在 1 秒的时间间隔内可以读取和检测 CPUSS_SYSREQ 的次数。

5.3　XRES_PULSE_100US

要想复位目标器件，主机将在 XRES 线路上生成一个低电平有效信号，它的最小脉冲宽度为 5 µs。在 HSSP 代码中，将在 XRES 引脚上生成宽度为 100 µs 的脉冲。

TimeoutCalc.c 文件定义了 `TestDelayHundredUs()`函数。

在该应用笔记宏提供的 C_Hssp_TimeoutCalc 项目内，该函数是通过 CyDelayUs() API 来实现的。通过取消注释 *TimeoutCalc.h* 中相应的行，可以将微控制器 (而不是 PSoC™) 作为主机编程器，并用该编程器执行该函数。

```
#define XRES_PULSE_100US  600
```

按照 Code Listing 3 提供的信息，修改 *TimeoutCalc.c* 中定义的 `TestDelayHundredUs()` 的代码。

要想计算此宏，请取消项目中 main.c 文件的 `TestDelayHundredUs()`函数，然后进行编程器件。使用示波器测量测试引脚的脉冲宽度。

Code Listing 3　延迟子程序

```
unsignedshort timestamp;


/* Make the pin low before start of the delay */
TESTPIN_OUTPUT_LOW;


/* For loop to introduce the 100 us delay
for(timestamp = 0; timestamp < XRES_PULSE_100US; timestamp++)
{
      /* Do Nothing */
```

计算 HSSP 超时参数

Code Listing 3　延迟子程序

```
}


/* Make the pin high after end of the delay */
TESTPIN_OUTPUT_HIGH;
```

A_HSSP_Programmer 和 B_HSSP_Pioneer 等项目使用 CyDelayUs () API 实现 100 µs 的延时。如果使用非 PSoC™的微控制器作为主机编程器，则必须在 *Timeout.h* 中定义 XRES_PULSE_100US 参数，并使用该参数 (位于 *Timeout.c* 文件中的 DelayHundredUs 函数内) 实现该延时。

对于电源循环编程，该延迟程序表示器件电源关闭的时间。对于电源循环编程，对该脉冲宽度没有任何要求。建议打开电源前，要经过 1 ms 的延迟时间。

6 接收 HSSP 编程数据的接口

DataFetch.c 和 *DataFetch.h* 中包含了各函数，用于提取编程到目标器件内的数据。

对于示例项目，编程数据存储在 PSoC™ 5LP 主机编程器的片上闪存中 *HexImage.c* 和 *HexImage.h* 文件内。为了执行 HSSP 操作，提取数据的子程序从 PSoC™ 5LP 闪存访问此编程数据。

然而，某些主机编程器可能没有用于存储 HSSP 编程数据的片上存储器。这时，编程器可以使用通信接口 (如 SPI、USB 或 UART) 获取编程数据。这样，应该适当修改 *DataFetch.c* 文件中的函数定义。

以下的参考示例介绍了需要对 `Hex_ReadRowData()`函数进行的修改。此外，您还可以对其他函数进行类似的修改。

源代码

Code Listing 4 用于获取闪存数据的源代码

```
void HEX_ReadRowData(unsignedshortrowCount, unsignedchar * rowData)
{
/* Maximum value of 'i' can be 256 */
unsignedshort i;

for(i = 0; i<BYTES_PER_FLASH_ROW; i++)
    {
rowData[i] = FlashData_HexFile[rowCount][i];
    }
}
```

已修改的代码

如果通过通信接口接收编程数据，则已修改的代码应与以下的代码相似。

Code Listing 5 用于获取闪存数据的已修改代码

```
void HEX_ReadRowData(unsignedshortrowCount, unsignedchar * rowData)
{
/* Maximum value of 'i' can be 256 */
unsignedshort i;

/* ADD WAITING CODE HERE FOR THE UART BUFFER TO GET THE FLASH DATA */

for(i = 0; i<BYTES_PER_FLASH_ROW; i++)
```

接收 HSSP 编程数据的接口

Code Listing 5　用于获取闪存数据的已修改代码

```
    {
rowData[i] = /* PLACE THE UART BUFFER ARRAY HERE */
    }
}
```

7 HSSP 时序验证

主机编程器必须符合目标器件编程的时序规范，以实现一个功能强大的 HSSP 操作。有关这些规范的详细信息，请参考**相关文档**一节列出的相应器件编程规范文档中“附录 D. SWD 接口的时序规范”部分。

主机编程器必须满足 SWD 接口和进入编程模式有关的时序参数。要想验证时序，请使用示波器捕获 SWDIO、SWDCK 和 XRES 信号。对于电源循环模式编程，需要监控器件电源 (而不是 XRES 引脚)。通过捕获到的波形，您可以使用编程规范所提供的相应值来验证时序参数。

8 电源循环模式编程

开始器件编程前，需要进行器件复位，以获取器件并进入编程模式。推荐在主机端通过切换器件的 XRES 引脚实现复位器件。但 PSoC™ 4000 系列中某些引脚数量少的器件没有 XRES 引脚，因而主机要通过切换 V_{dd} 引脚来复位器件 (电源循环模式)。本应用笔记所提供的全部项目都使用了 XRES 编程模式，因为开发板上的 PSoC™ 5LP 主机编程器没有用于切换 PSoC™ 4 器件电源的硬件连接。因此，将项目移植到您的主机编程器中，并添加电源循环模式编程时，需要对项目进行以下修改，并要注意以下内容:

1. 需要更改在 **DEVICE_ACQUIRE_TIMEOUT** 一节中所介绍的 DEVICE_ACQUIRE_TIMEOUT 值，使之反映电源循环模式编程中所需的 30 ms 超时时间。因此，如果使用 2 ms 的 XRES 模式获取时序，DEVICE_ACQUIRE_TIMEOUT 被定义为 20，那么使用 30 ms 的电源循环模式获取时序时，该定义应改为 300。
2. 为了切换器件电源，从主机端切换 XRES 引脚的有关函数定义需要进行相应修改。这些函数是: `SetXresHigh()`和 `SetXresLow()`。
3. 如果使用主机处理器的一个 I/O 引脚直接给 PSoC™ 4 器件供电，进而通过主机切换电源，那么需要保证 I/O 引脚能够提供器件操作所需电流，如相应的 PSoC™ 4 器件数据手册中所指定的内容。如果 I/O 引脚不能提供所需电流，那么输出电压 (V_{OH}) 通常会下降，并且可能下降到 PSoC™ 4 器件工作电压以下。这种现象会引起编程故障。通过观察示波器上的电源电压，可以发现这种电压下降现象。

9 测试示例项目

HexImage.h 项目文件定义了下面各参数: CY8C40xx_FAMILY、CY8C4xxxM_FAMILY、CY8C4xx7_BL_FAMILY、CY8C4xx8_BL_FAMILY、CY8C4xxxL_FAMILY、CY8C40xxS_FAMILY、CY8C41xxS_FAMILY、CY8C41xxS_PLUS_FAMILY 和 CY8C41xxPS_FAMILY。如果相应系列的 Hex 文件解析应用用于生成 *HexImage.c* 和 *HexImage.h* 文件，这些参数将自动被设置为 1。

9.1 对于 CY8CKIT-038 PSoC™ 4 开发套件

要想测试 **CY8CKIT-001 DVK** 中 PSoC™ 4 处理器模块 (**CY8CKIT-038**) 上的 HSSP 项目 (其中使用 PSoC™ 5LP 作为主机)，请使用本应用笔记附带的 A_Hssp_Programmer 项目。对 **CY8CKIT-050 DVK** 中 PSoC™ 5LP 的项目进行编程。**Figure 4** 提供了将主机与目标器件相连的引导。按下 SW2，开始 HSSP 操作。

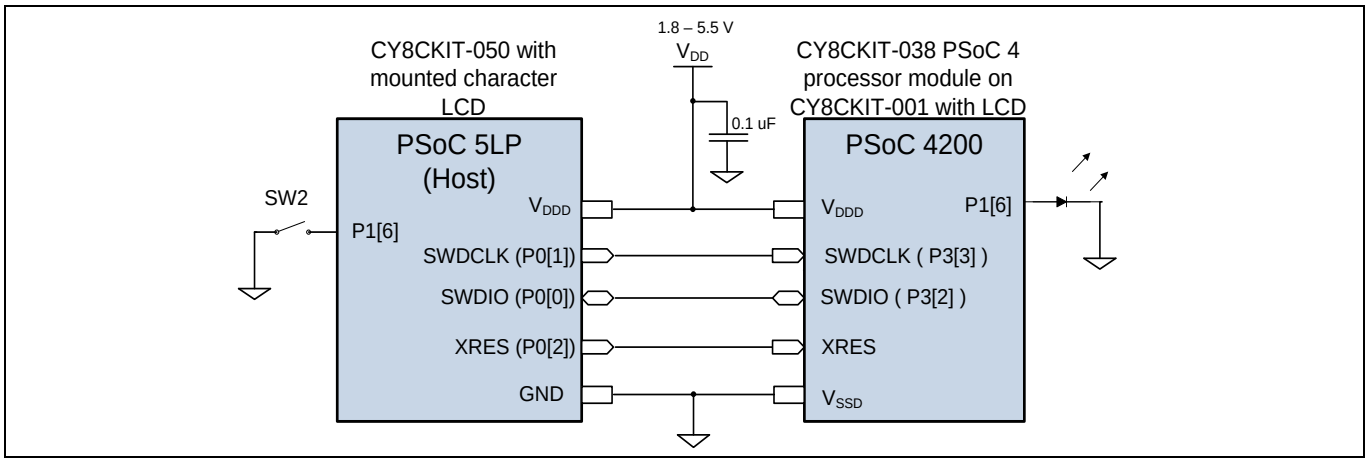


Figure 4 主机编程器和目标器件之间的连接

编程操作成功后，本项目中的'hex'文件会以 1 Hz 频率将 PSoC™ 4 的引脚 P1[6]切换，并在 **CY8CKIT-001 DVK** 上安装的字符 LCD 上显示"PSoC Programmed"。按下 PSoC™ 5LP 主机上的 SW2，开始编程操作。如果编程操作成功，引脚 P1[6]将开始切换，并在字符 LCD 上显示信息。如果编程操作失败，则 PSoC™ 5LP 将在 LCD 上显示错误的原因 (该 LCD 安装在 PSoC™ 5LP 套件上)。

Note: *如果您使用其他主机编程器，请按照**将 HSSP 应用移植到主机编程器**一节进行修改源代码。然后，通过进行基本连接 (如 **Figure 3** 所示) 来测试项目。*

9.2 对于具有板上 PSoC™ 5LP 编程器 (KitProg) 的套件

为了对 **Table 10** 中所列的套件测试 HSSP 项目，请使用本应用笔记附带的 B_Hssp_Pioneer 项目。使用该套件时，不需要外部主机微控制器；PSoC™ 5LP 将作为板上微控制器使用。

板上 PSoC™ 5LP 具有 Bootloader 固件，能够通过 USB 下载并运行新的可启动加载应用程序。因此，HSSP 项目构建为可启动加载项目，您可以通过 USB 将 *B_Hssp_Pioneer.cyacd* 项目下载到 PSoC™ 5LP，以将 PSoC™ 5LP 用作 HSSP 主机编程器。

该项目使用一个 USB 到 UART 的组件，使各编程输出显示在超级终端 (HyperTerminal) 上。这是一个用于串行通信的标准程序。

该项目要求实现 **Table 10** 所列出的更改，以便在 CY8CKIT-040、CY8CKIT-042、CY8CKIT-044 和 CY8CKIT-042-BLE 上工作。请注意，相应更改应按照项目的 *RegisterDefines.h* 文件进行。

测试示例项目

Table 10　　各种套件的引脚分配

主机引脚	CY8CKIT-040	CY8CKIT-042	CY8CKIT-043	CY8CKIT-044	CY8CKIT-046	CY8CKIT-042-BLE
Pin_SWDCK	P12[3]	P2[1]	P12[3]	P12[3]	P12[3]	P12[3]
Pin_SWDIO	P12[2]	P2[0]	P12[2]	P12[2]	P12[2]	P12[2]
Pin_XRES	P12[4]	P2[4]	P12[4]	P12[4]	P12[4]	P12[4]

请按照以下各步骤对本项目进行测试:

1. **准备 B_Hssp_Pioneer 项目:**
   a) 使用 **HexFile Parser** 为您的目标 PSoC™ 4 器件生成包含了编程数据的文件 (*HexImage.c*、*HexImage.h*)。
   b) 使用所生成的文件替换掉现有的 *HexImage.c* 和 *HexImage.h* 文件。
   c) 双击 *TopDesign.cysch* 中的 Bootloadable 组件。
   d) 选择 **Dependencies** 选项卡，然后点击与 Bootloader HEX 文件相关的浏览按键，如 **Figure 5** 所示。
   e) 导航到本应用笔记所提供的 PSoC™ 5LP Bootloader Files > KitProg 文件夹，然后选择 *KitProg_Bootloader.hex*。

   当您选择 *Bootloader HEX* 文件时，Bootloader ELF 文件将被自动生成。

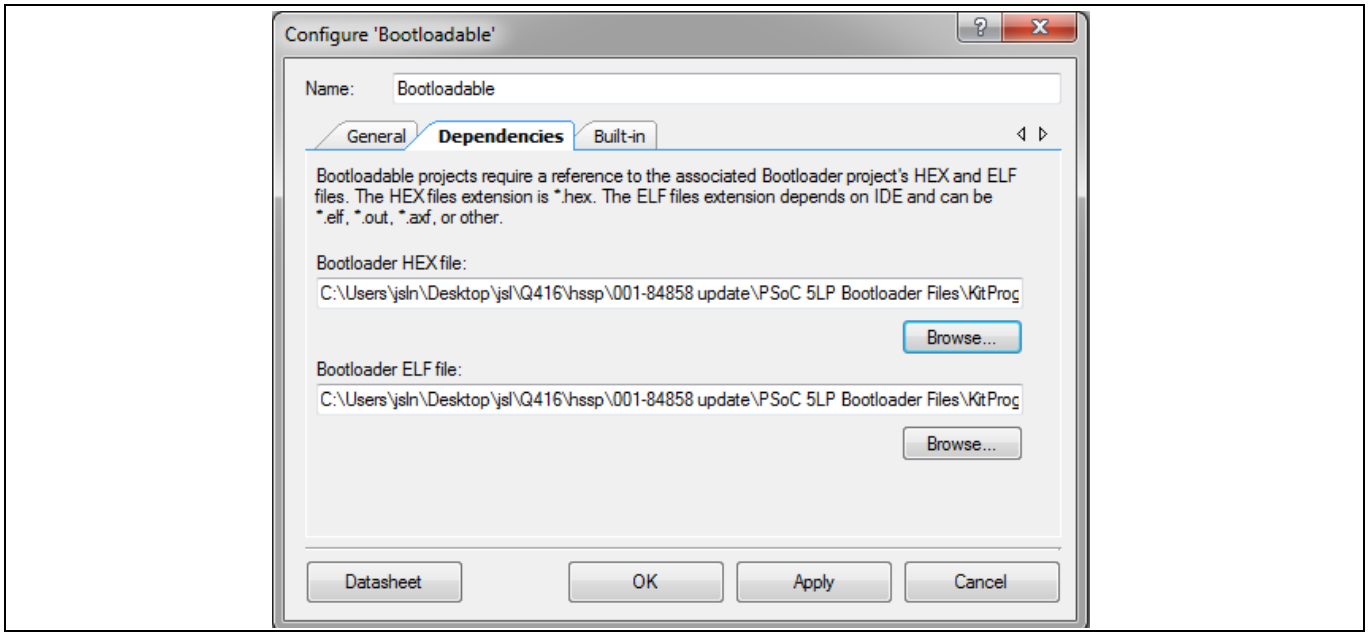


Figure 5　　用于 KitProg 的 Bootloadable 依赖属性

f) 构建 B_Hssp_Pioneer 项目。

2. **进入 Pioneer 套件中的 PSoC™ 5LP Bootloader:**
   a) 断开 USB 电缆连接，以关闭电源。
   b) 按住套件上的复位开关 (SW1)，同时插入 USB 电缆。

   状态 LED 开始闪烁，表示 PSoC™ 5LP 已进入 Bootloader 模式。**Figure 6** 显示了 PSoC™ 4 Pioneer 套件以及状态 LED 和复位开关的位置。



Figure 6 PSoC™ 4 Pioneer 套件

3. **将 HSSP 项目启动加载到 PSoC™ 5LP 内:**
   a) 打开 Bootloader 主机工具。选择 PSoC™ Creator 中的 **Tools** > **Bootloader Host**。
   b) 点击 **Filter**，然后勾选 **Show USB Devices** 复选框。
   c) 在 **PID** 字段中，输入‘0xF13B’ (请查看 **Figure 7**)。
   d) 在端口列表中，PSoC™ 5LP Bootloader 列为 USB 人机接口器件。
   e) 在 GUI 中点击 **Open**，然后在下面的路径选择 *B_Hssp_Pioneer.cyacd* 文件:

   ..\ AN84858 \ B_Hssp_Pioneer.cydsn \ CortexM3 \ARM_GCC_441 \Debug \B_Hssp_Pioneer.cyacd

   f) 点击 **Program** 按键，以将该文件启动加载到 PSoC™ 5LP 内。

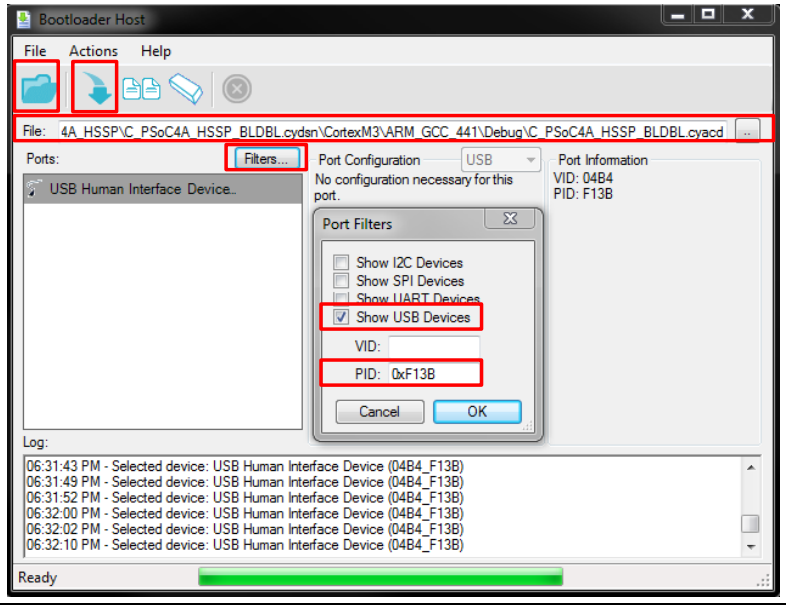


Figure 7 Bootloader 主机

4. **给套件安装 USBUART 驱动程序:**
   a) 断开，然后重新插入 USB 电缆。
   b) 如果 Windows 系统无法安装 USBUART 驱动程序，请打开 **Device Manager**。请确保 **Other devices** 列表中具有 **USBUART**。

测试示例项目

c) 双击 **USBUART** 来打开它的属性。

d) 点击 **Update Driver**。

e) 选中 **Browse my computer for driver software** 选项。

f) 驱动程序位于附带项目的‘USBUART’文件夹中。导航到保存本应用笔记的附加项目的位置，并在 *USBUART driver* 文件夹中选择驱动程序。

g) 点击 **Next**。由于驱动程序文件没有被数字签名，因此 Windows 系统会生成一个警告。忽略该警告并继续操作。现在，USBART 驱动程序已安装在您的电脑上。

5. 使用超级终端开始 HSSP 编程:

a) 在您的电脑上打开超级终端。如果您的电脑上没有该终端，可以在网上下载适用于串行通信的任何终端应用。

b) 在 UART 配置窗口中，分别将波特率设为 9600、数据位数为 8、停止位数为 1、 奇偶校验为‘No Parity’ (无奇偶校验) 以及硬件控制为‘None’ (无)。

c) 按下键盘上的任何字母数字按键，开始编程操作。如果编程操作成功，终端应用上会提示‘HSSP Success’信息 (请查看 **Figure 8**)。

d) 如果编程操作失败，终端应用上会显示您可以用来调试项目的步骤和错误代码 (请查看 **Figure 8**)。

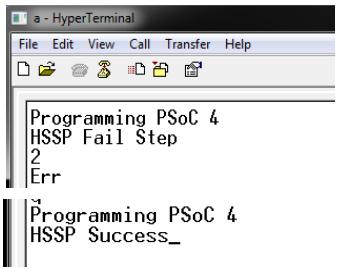


Figure 8 终端显示

6. 使用套件固件对 PSoC™ 5LP 进行重新编程:

a) 打开 PSoC Programmer。

b) 按照第二步进入 Bootloader 模式。

c) 进入 **Utilities** 选项卡，然后点击 **Upgrade Firmware**，以便使用原始的套件固件重新编程 PSoC™ 5LP (**Figure 9**)。

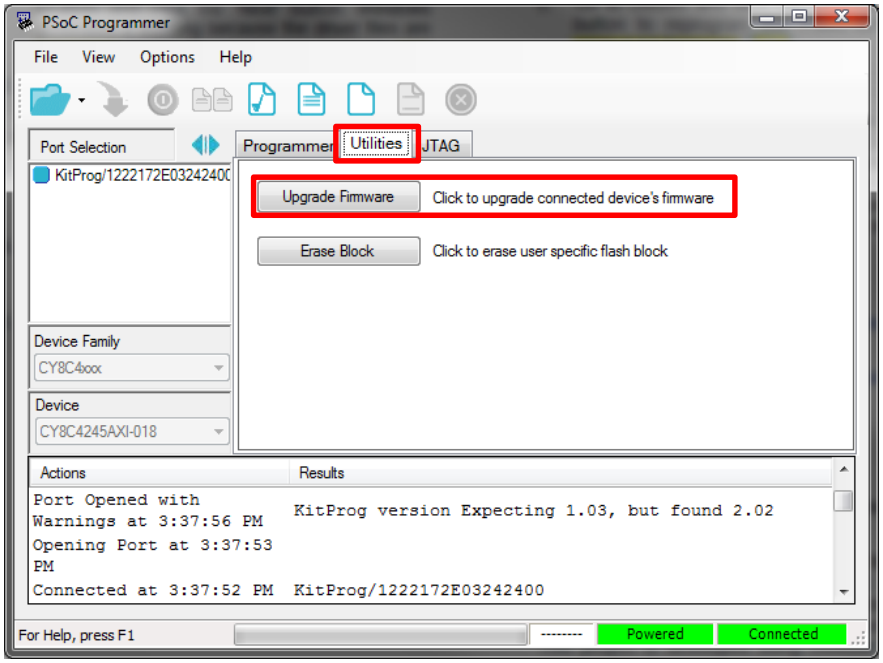


Figure 9　使用 PSoC™编程器升级固件

9.3　对于具有板上 PSoC™ 5LP 编程器的套件 (KitProg2)

为了对 **Table 10** 中所列的套件测试 HSSP 项目，请使用本应用笔记附带的 B_Hssp_Pioneer 项目。使用该套件时，不需要外部主机微控制器；PSoC™ 5LP 将作为板上微控制器使用。

板上 PSoC™ 5LP 器件具有 Bootloader 固件，能够通过 USB 下载并运行新的可启动加载应用程序。因此，HSSP 项目构建为可启动加载项目，您可以通过 USB 将 *B_Hssp_Pioneer.cyacd* 项目下载到 PSoC™ 5LP，以将 PSoC™ 5LP 用作 HSSP 主机编程器。

该项目使用一个 USB 到 UART 的组件，使各编程输出显示在超级终端 (HyperTerminal) 上。这是一个用于串行通信的标准程序。

为了与 CY8CKIT-041-40xx 套件工作，该项目要求进行一些更改，如 **Table 10** 所示。请注意，相应更改应按照项目的 *RegisterDefines.h* 文件进行。

Table 11　各种套件的引脚分配

主机引脚	CY8CKIT-041-40xx	CY8CKIT-149	CY8CKIT-147
Pin_SWDCK	P12[3]	P12[3]	P12[3]
Pin_SWDIO	P12[2]	P12[2]	P12[2]
Pin_XRES	P12[4]	P12[4]	P12[4]

请按照以下各步骤对本项目进行测试:

1. **准备 B_Hssp_Pioneer 项目:**
   a) 使用 **HexFile Parser** 为您的目标器件生成包含了编程数据的文件 (*HexImage.c*、*HexImage.h*)。
   b) 使用所生成的文件替换掉现有的 *HexImage.c* 和 *HexImage.h* 文件。
   c) 双击 TopDesign.cysch 中的 Bootloadable 组件。

d) 选择 **Dependencies** 选项卡，然后点击与 Bootloader HEX 文件相关的 **Browse**，如 **Figure 10** 所示。

e) 导航到本应用笔记所提供的 **PSoC 5LP Bootloader Files** > **KitPog2** 文件夹，然后选择 *KitProg2_Bootloader.hex*，用于 Bootloader HEX 文件字段。

当选择 Bootloader HEX 文件时，Bootloader ELF 文件将被自动生成。

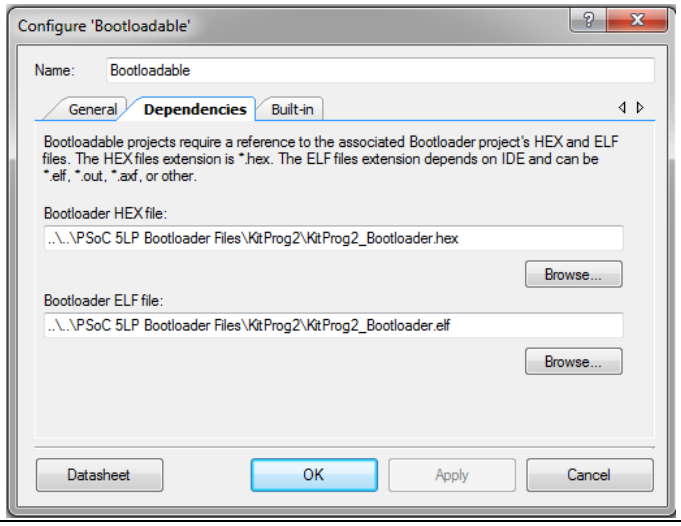


Figure 10　用于 KitProg2 的 Bootloadable 依赖属性

f) 构建 B_Hssp_Pioneer 项目。

2. 进入 Pioneer 套件中的 PSoC™ 5LP Bootloader:

a) 断开 USB 电缆连接，以关闭电源。

b) 按住套件上的 SW3，同时插入 USB 电缆。

LED2 开始闪烁，表示 PSoC™ 5LP 已进入 Bootloader 模式。**Figure 11** 显示了 PSoC™ CY8CKIT-041-040xx 套件和 LED2 和 SW3 的位置。



Figure 11　CY8CKIT – 041-040xx

3. 将 HSSP 项目启动加载到 PSoC™ 5LP 内:

a) 打开 Bootloader 主机工具。在 PSoC™ Creator 中依次选择 **Tools** > **Bootloader Host**。

b) 点击 Filter 按键，然后选择 **Show USB Device** 复选框。

测试示例项目

c) 在 PID 字段中输入‘0XF146’ (请参阅 **Figure 12**)。

d) 在端口列表中，PSoC™ 5LP Bootloader 列为 USB 人机接口器件。

e) 在 GUI 中点击 **Open**，然后在下面的路径选择 *B_Hssp_Pioneer.cyacd* 文件:

..\ AN84858 \ B_Hssp_Pioneer.cydsn \ CortexM3 \ARM_GCC_441 \Debug \B_Hssp_Pioneer_2.cyacd

f) 点击 **Program** 按键，以将该文件启动加载到 PSoC™ 5LP 内。

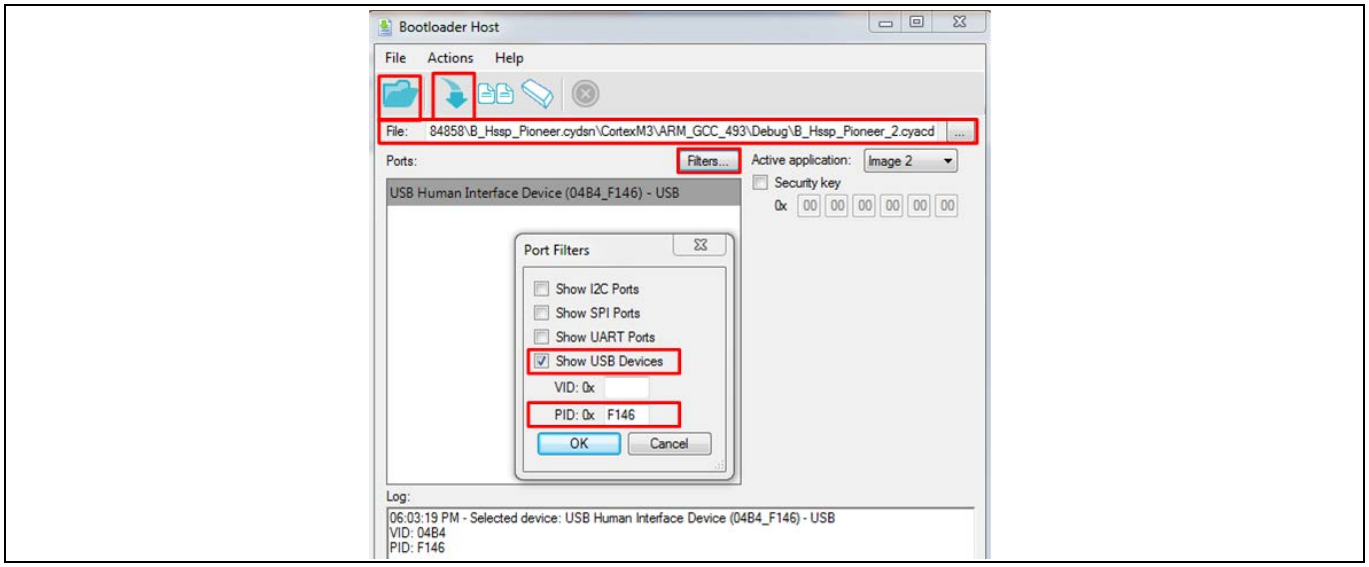


Figure 12 Bootloader 主机应用程序

4. 给套件安装 USBUART 驱动程序:

a) 断开，然后重新插入 USB 电缆。

b) 如果微软无法安装 USBUART 驱动器，请执行下面各步骤:

a. 打开 **Device Manager**。请确保 **Other devices** 列表中具有 **USBUART**。

b. 双击 USBUART 来打开它的属性。

c. 点击 **Update Driver**。

d. 选中 **Browse my computer for driver software** 选项。

e. 导航到保存本应用笔记的附加项目的位置，并在 *USBUART driver* 文件夹中选择驱动程序。

f. 然后点击 **Next** 按键。由于驱动程序文件没有被数字签名，因此 Windows 系统会生成一个警告。忽略该警告并继续操作。现在，USBART 驱动程序已安装在您的电脑上。

5. 使用超级终端开始 HSSP 编程:

a) 在您的电脑上打开超级终端。如果您的电脑上没有该终端，可以在网上下载适用于串行通信的任何终端应用。

b) 在 UART 配置窗口中，分别将波特率设为 9600、数据位数为 8、停止位数为 1、 奇偶校验为‘No Parity’ (无奇偶校验) 以及硬件控制为‘None’ (无)。

c) 按下键盘上的任何字母数字按键，开始编程操作。如果编程操作成功，终端应用上会提示‘HSSP Success’信息 (请查看 **Figure 13**)。

d) 如果编程操作失败，终端应用上会显示您可以用来调试项目的步骤和错误代码 (请查看 **Figure 13**)。

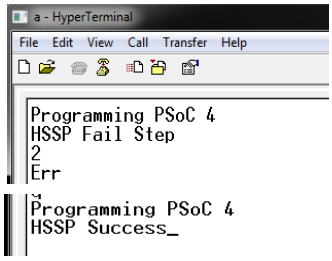


Figure 13 终端显示

6. **使用套件固件对 PSoC™ 5LP 进行重新编程:**

   a) 打开 PSoC Programmer。

   b) 按照第二步进入 Bootloader 模式。

   c) 进入 **Utilities** 选项卡，然后点击 **Upgrade Firmware** ，以便使用原始的套件固件重新编程 PSoC™ 5LP (**Figure 14**)

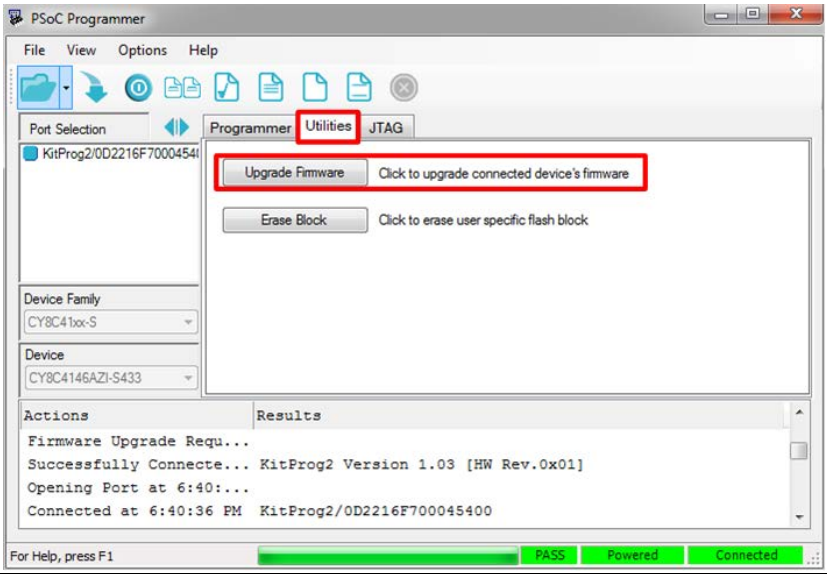


Figure 14 使用 PSoC™编程器升级固件

10 解决有关 HSSP 问题的提示和技巧

根据系统对主机处理器的要求，将 HSSP 代码从代码示例中所用的 PSoC™ 5LP 主机内移植到您自己的处理器架构内可能比较复杂。本节内容有助于解决您在硬件平台上进行开发 HSSP 应用时遇到的最常见问题。

- **硬件设置验证**: 首先要保证硬件连接符合 HSSP 操作的要求。这些要求包括: 主机处理器与目标 PSoC™ 4 器件之间的引脚准确连接；PSoC™ 4 的所有电压域均被供电；主机 SWD 引脚驱动模式设置得到准确配置。有关硬件连接和配置的详细信息，请参考- **编程规范**-中相应器件编程规范内的“物理层”一节。
- **时序验证**: 将主机 PSoC™ 5LP 代码移植到您的主机处理器内时，请确保代码中所使用的超时参数已经得到修改，并符合主机处理器的代码超时。有关移植代码期间对超时参数进行修改的详细信息，请参考 **计算 HSSP 超时参数**一节。

HSSP 编程的第一步 (获取器件) 与进入编程模式之间存在较严格的时序要求。其中，一个重要的要求是确保 SWDCK 时钟线的频率最低为 1.5 MHz，从而可以满足获取时序的要求。请保证主机处理器内不会发生任何中断事件，因为这些事件会影响主机处理器完成“获取器件”步骤中代码执行时间。请确保主机处理器满足相对应的器件编程规范文档中“第一步 — 获取芯片”所列出的所有时序要求。

- **HSSP 算术验证**:
  - 移植 HSSP 代码时，如果修改了 **Figure 1** 中 SWD 数据包层的各个文件，则需要保证 SWD 数据包格式与器件编程规范中“串行线调试 (SWD) 格式”一节所描述的格式相同。
  - 经过英飞凌认证的编程器 (如 MiniProg3、KitProg) 可以用于验证和调试 **Figure 1** 中的“擦除闪存”、“编程闪存”等步骤。例如，要想检查主机处理器是否擦除了整个闪存，可用 MiniProg3 编程器和 PSoC™ 编程器 GUI 中的“读取”选项来验证整个闪存数据是否为零。PSoC™编程器 GUI 中的“校验和”选项可用于验证编程到器件内的闪存数据校验和是否与作为 GUI 输入文件的 Hex 文件校验和相匹配。此外，PSoC™编程器 GUI 中的“Patch Image”选项可用于确定导致器件闪存内容与 Hex 文件数据不匹配的闪存行数量。

11 总结

开发 PSoC™ 4 器件的在线编程解决方案时，HSSP 应用极为有用。它为 PSoC™ 4 器件提供了一种既廉价又可靠的编程方法，其中使用板上嵌入式微控制器作为主机编程器。通过使用本应用笔记所提供的可移植模块化 C 代码，可缩短开发 HSSP 应用的时间。

12 相关文档

12.1 应用笔记

- AN73054 — 使用外部微控制器对 PSoC™ 3 / PSoC™ 5LP 进行编程 (HSSP)
- AN44168 — 使用外部微控制器对 PSoC™ 1 器件进行编程 (HSSP)

12.2 编程规范

- CYBL10x6x、CY8C4127_BL、CY8C4247_B: 编程规范
- CY8C4XXXM、PSoC™ 41xxS、PSoC™ 41xxS Plus、PSoC™ 40xxS、PSoC™ 4100PS 编程规范
- CY8C41XX、CY8C42XX 编程规范
- CY8C4000 编程规范

12.3 架构技术参考手册

- PSoC™ 4 系列: PSoC™ 4 架构技术参考手册 (TRM)
- PSoC™ 4100 和 4200 系列: PSoC™ 4 架构技术参考手册 (TRM)
- PSoC™ 4100M/4200M 系列: PSoC™ 4 架构技术参考手册 (TRM)
- PSoC™ 41X7_BLE/42X7_BLE 系列: PSoC™ 4 BLE 架构技术参考手册 (TRM)
- PSoC™ 4100S 和 PSoC™ 4100S Plus: PSoC™ 4 架构技术参考手册 (TRM)
- PSoC™ 4000S: PSoC™ 4 架构技术参考手册 (TRM)
- PSoC™ 4100PS: PSoC™ 4 架构技术参考手册 (TRM)

12.4 网页

- 通用的 PSoC™编程

13 随附的项目

AN84858.cywrk: 该工作区包含三个项目，用于演示 HSSP 应用。

- A_Hssp_Programmer: 这是 PSoC™ 4 的 HSSP 应用项目。它使用 PSoC™ 5LP 器件作为主机编程器，以便对目标 PSoC™ 4 器件进行编程。该项目使用模块化 C 代码开发，以根据**相关文档**一节中所列出的相应器件编程规范文档中所述的步骤对器件进行编程。
- B_Hssp_Pioneer: 这是 PSoC™ 4 的 HSSP 应用项目，用于在 PSoC™ 4 Pioneer 套件上进行测试。它使用板上 PSoC™ 5LP 编程器对 PSoC™ 4 器件进行编程。
- C_Hssp_TimeoutCalc: 该项目用于计算 PSoC™ 4 HSSP 项目中所用的 timestamp 参数。

另外，本应用笔记还提供了 C#应用，用于提取 Hex 文件中的信息。

- **Hex 文件解析应用**: 该 C#应用从 Hex 文件提取所需信息，并将它解析为*.c/h* 文件。此文件存储于微控制器的闪存内，用于直接访问编程数据。

14 附录 A. Hex 文件解析器应用

用于编程 PSoC™ 4 的数据以 hex 文件 (*.hex*) 格式存储。有关该格式的详细内容，请在**相关文档**一节所列出的相应器件编程规范文档中查看“附录 B. Intel Hex 文件格式”部分。

Hex 文件格式不同于主机编程目标器件时所用的格式。这个文件由 PSoC™ Creator 软件生成，其中的编程数据和元数据均以十六进制格式存储。元数据包括 hex 文件纪录类型、扩展的线性地址数据等信息。元数据用于将编程数据分为闪存代码区域数据、闪存配置区域数据、闪存保护数据等类型。

使用 Visual Studio 开发环境开发的 C#应用可以解析 hex 文件，并生成*.c*、*.h* (即 *HexImage.c*、 *HexImage.h*) 文件。这些文件仅存储 hex 文件的编程数据。编程数据在 *HexImage.c* 和 *HexImage.h* 文件中以常量阵列的形式存储。

本应用笔记提供了 C#应用以及其源代码。要想使用 C#应用，您必须在电脑上安装‘.NET’框架版本 3.5 或更高版本。

Note: *为 PSoC™ 4000、PSoC™ 4100/4200、PSoC™ 4100M/4200M、PSoC™ 4xx7_BLE、PSoC™ 4xx8_BLE、PSoC™ 4xxxL、PSoC™ 40xxS、PSoC™ 41xxS、PSoC™ 41xxS Plus 和 PSoC™ 41xxPS 器件系列提供了单独的 Hex 文件解析应用。*

C#应用名称: Hex 文件解析

开发环境: Microsoft Visual Studio Express 2013 (版本 12.0.31101.00 更新 4)

源代码: 有关详细信息，请查看 C#源项目一节。通过下载和安装 Microsoft 免费提供的 Visual Studio 2013 Express Edition 工具，可以查看和编辑该项目的源代码。只要右击 **Solution Explorer** 窗口中的 *Form1.cs* 文件，然后选择 **View Code** 选项，便可打开 *Form1.cs* 文件中的源代码。

14.1 使用 Hex 文件解析应用

可在“C#应用”文件夹中寻找九组解析应用。根据您正在使用的 PSoC™ 4 系列，选择适当的应用程序。

在附加‘*.zip*’文件的 *C# Application\HexFile Parser.exe* 文件夹中，请打开 Hex 文件解析器应用的可执行文件。在此，将弹出一个 GUI 屏幕，如 **Figure 15** 所示。

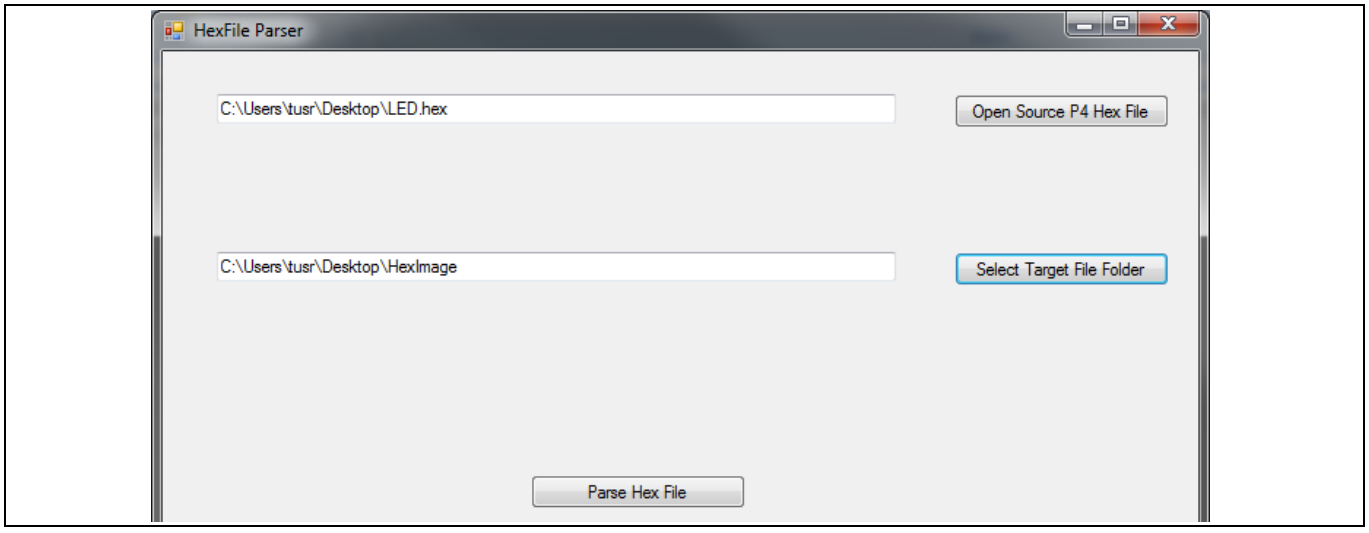


Figure 15 Hex 文件解析器应用

1. 请点击 **Open Source P4 Hex File** 并导航至需要进行编程的 hex 文件所在的位置来做出选择。

2. 选择经过解析的‘*.c* 和*.h*’文件 (即 *HexImage.h*、*HexImage.c*) 所在的目标文件夹位置。点击 **Select Target File Folder** 来选择文件夹位置。

Note: *请确保目标文件夹中不包含名为 HexImage.h 或 HexImage.c 的文件。如果有，请删除它们，或者选择新的文件夹中。如果目标文件夹的位置已经包含了具有同样名称的文件，那么会向用户提示信息。*

3. 选中 Hex 文件和目标文件夹位置后，点击 **Parse Hex File**，以生成*.c* 和*.h* 文件。完成解析后，将显示一条信息。

14.2 将所生成的文件添加到 PSoC™ Creator 示例项目中

要想将所生成的 *HexImage.c 和 HexImage.h* 文件添加到本应用手册所提供的 PSoC™ Creator 示例项目中，请按照本节中所述的步骤执行操作。如果您需要使用 PSoC™ 5LP 作为主机编程器来将 hex 文件编程到目标器件中，则必须执行这些步骤。这些步骤适用于 PSoC™ 4 HSSP 的 A_Hssp_Programmer 和 B_Hssp_Pioneer 项目。

1. 在 GUI 中选择目标文件文件夹的位置，如 **Figure 15** 所示。这就是项目的 *main.c* 文件所在的文件夹。
2. 在上述位置生成 *HexImage.c* 和 *HexImage.h* 文件后，点击 Workspace Explorer 窗口中的 **Add Existing Item** 选项，将那些文件添加到 PSoC™ Creator 的项目工作区中，如 **Figure 16** 所示。

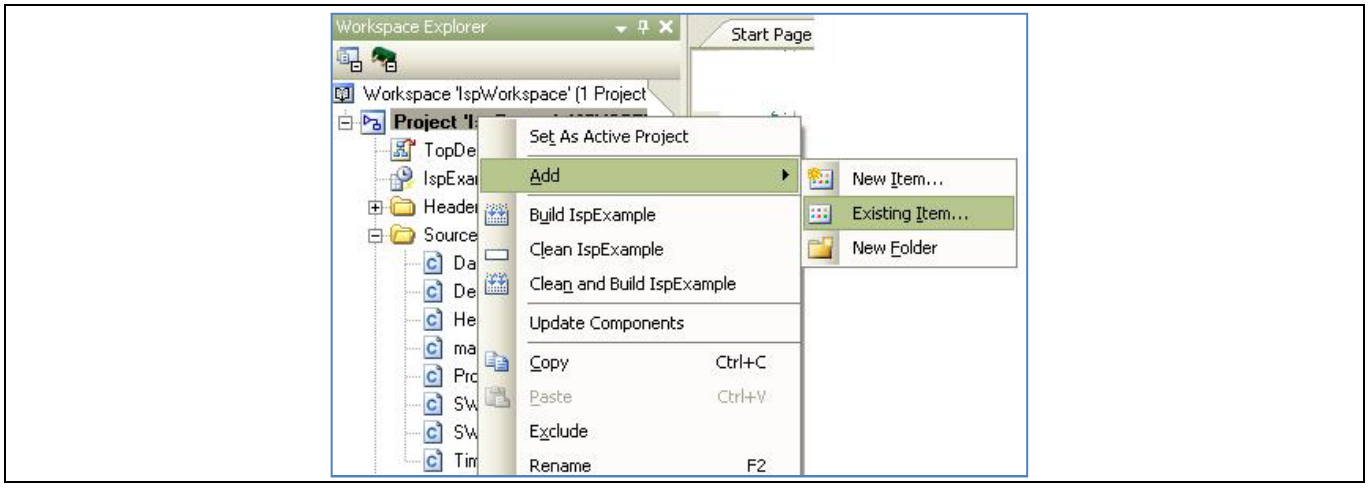


Figure 16 将文件添加到 PSoC™ Creator 项目中

3. 选择这些文件后，Workspace Explorer 窗口将显示，如 **Figure 17** 所示。

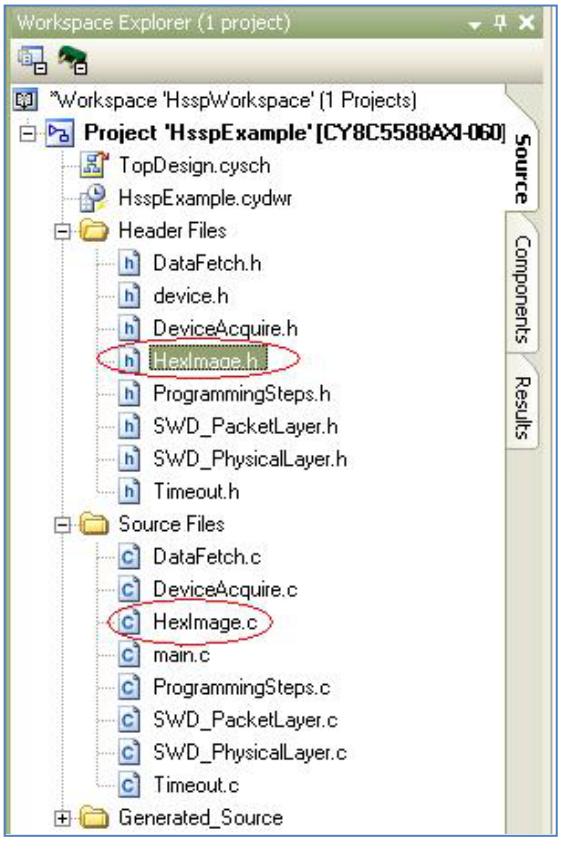


Figure 17　项目工作区显示了 *HexImage.c* 和 *HexImage.h* 文件

如果您需要使用另一个 hex 文件的其他相应文件取代这些文件，请右击来删除工作区 Explorer 中和项目文件夹中现有的文件。然后，请进行步骤 **1** 至步骤 **3** 的操作，以将新文件添加到项目中。

14.2.1　对于 PSoC™ 4xxxL/ PSoC™ 41x8_BLE/ 42x8_BLE/ 41XXS Plus 器件系列，请使用 *HexImage.c*、*HexImage.h* 文件:

1. HSSP 代码示例使用 PSoC™ 5LP 器件作为主机编程器，来编程目标 PSoC™ 4 器件。由于 PSoC™ 5LP 的 HSSP 算术需要使用它的 256 KB 大小闪存中的一部分，因此，闪存容量大的器件 (如 256 KB PSoC™ 4xxxL/PSoC™ 41x8_BLE/42x8_BLE 器件系列) 的整个 Hex 文件不能存储在主机 PSoC™ 5LP 的闪存内。
2. 在 B_HSSP_Pioneer 项目中，PSoC™ 41xxS Plus 器件的全部 128 KB 无法存储在 CY8CKIT-159 的板载 PSoC™ 5 中，因为它使用具有双应用映像引导加载功能的 KitProg2。
3. 为了满足代码大小，已经移除了 *HexImage.c* 和 *HexImage.h* 文件内最后几个闪存行中的编程数据。相应的逻辑也包括在 *DataFetch.c* 文件的 `HEX_ReadRowData(…)`函数内。下面是所进行的修改。要想查看正确变更的详细信息，请参见项目代码的内容。
4. 在 *HexImage.h* 文件中，通过下面的编译，可减少 256 个数组声明中的闪存行:

   ```
   extern unsigned char const
   flashData_HexFile[(NUMBER_OF_FLASH_ROWS_HEX_FILE -
   256)][FLASH_ROW_BYTE_SIZE_HEX_FILE];
   ```

   由于已经移除了 256 行，项目的代码大小完全适合所有目标器件的主机 PSoC™ 5LP 闪存存储器的容量。

5. 在 *HexImage.c* 文件中，随着 *HexImage.h* 文件中数组声明的改变，`flashData_HexFile[][]`数组定义中最后 256 个闪存行也相应被移除。

6. 在 DataFetch.c 文件中 HEX_ReadRowData(...)函数的定义内，如果闪存行号即为目标器件的最后 256 行，则整个闪存行数据将被加载为零。

15 附录 B. SROM 请求的状态代码

编程相关的所有操作均通过调用 “System Call”函数实现。在特权操作模式下这些操作在 SROM 外执行。

用户不能读取或修改 SROM 代码。将函数操作码、函数参数写入特定寄存器内，然后请求 SROM 执行函数。这样 DAP 或 Cortex-M0 CPU 将调用系统函数。根据函数操作码，SROM 将从它的存储器中执行相应的系统调用，并在寄存器中更新函数的执行状态。为了获取函数执行的成功/失败结果，DAP 或 CPU 应当读取该状态寄存器。

当 CPUSS_SYSREQ 寄存器中的 SROM_SYSREQ_BIT (位 31) 和 SROM_PRIVILEGED_BIT (位 28) 被清除时，则表示已完成系统调用。通过读取 CPUSS_SYSARG 寄存器，可以检查系统调用操作是否成功。

若从寄存器上读取的 32 位值为 0xAXXXXXXX (其中 X 表示无需关注的值)，则表示系统调用操作已成功。如果该值为 0xF00000YY 格式，则表示系统调用失败，并且 YY 会指出失败的原因。**Table 12** 显示了错误代码列表。

更多有关信息，请参考**相关文档**中列出的相应 PSoC™ 4 架构技术参考手册 (TRM) 中“非易失性存储器编程”一节。

Table 12 错误状态代码与失败的原因

状态代买 (CPUSS_SYSARG 寄存器内的 32 位值)	说明
0xAXXXXXXX	成功。“X”表示“无需关注”的值。除非 API 直接向 CPUSS_SYSARG 寄存器返回参数，否则该值包含了由 SROM 返回的 0。
0x F0000001	无效的芯片保护模式: 在当前的芯片保护模式下，不能使用该 API。
0x F0000003	无效的页锁存地址: 表示页锁存缓冲区中的地址超出了边界，或者为页面地址提供的大小过大。
0x F0000004	无效的地址: 所提供的行号或字节地址不处于可用的存储器范围内。
0x F0000005	受保护的行: 所提供的行号是一个受保护的行。
0x F0000006	SRAM 地址无效: SRAM 地址超出边界。
0x F0000007	恢复过程已完成: 所有无阻塞 API 操作已完成。不能调用恢复的 API，直到执行下一个非阻塞 API 为止。
0x F0000008	挂起恢复: 启动了一个无阻塞 API。调用任何其他 API 之前，必须通过调用一个恢复 API 来完成此操作。
0x F0000009	正在进行系统调用: 正在进行一个恢复的或无阻赛的 API。在尝试进行下一个恢复过程之前，必须触发 SPC ISR。
0x F000000A	零校验和失败: 所计算的校验和并非零值。
0x F000000B	无效的操作码: 操作码不是一个有效的 API 操作码。
0x F000000C	主操作码失配: 所提供的操作码与 key1 和 key2 不匹配。
0x F000000E	无效的起始地址: 起始地址大于所提供的结束地址。
0xF0000012	无效的闪存时钟: 发生写/擦除操作前，CY8C40xx 系列器件必须将 IMO 的频率设置为 48 MHz，并将 IMO 时钟设置为高速时钟源。

16 附录 C. HSSP 错误状态寄存器的位字段定义

HSSP 错误状态寄存器包含了当前 HSSP 操作的状态。HSSP 应用中的任何顶层步骤返回失败状态时，将从主应用代码调用 `ReadHsspErrorStatus()`函数，以获取错误信息的详情。该函数将返回该寄存器中的内容。欲了解此函数所返回的位字段，请查看 **Figure 2**。

为了能成功执行 HSSP 操作，该 8 位寄存器中的所有位 (位 0 除外) 必须为 0。设置‘位 0’表示 SWD 数据包收到了一个“OK ACK”响应，如 **Table 13** 所示。

16.1 位[2:0] — SWD 确认响应 (SWD ACK [2:0])

这是对 SWD 数据包的 3 位确认响应，该响应由目标器件发送至主机编程器。**Table 13** 列出了所有 ACK 代码。

Table 13 SWD ACK 响应代码

ACK[2:0]	ACK 响应含义	说明
001	OK (成功)	这表示前一个 SWD 数据操作已成功完成。
010	WAIT (等待)	该错误代码表示目标器件已经连续返回 5 个 WAIT ACK 响应。
100	FAULT (故障)	该错误代码指示: 在发送前一个 SWD 写入数据包的过程中，主机所发送的 4 字节数据包中有一个奇偶校验错误。
所有其他代码	未定义的代码	将它视为 FAULT 响应。

除“OK ACK”响应外，所有响应要求主机中止 HSSP 操作，同时从第一步重新开始。至于 OK ACK 响应，不应设置状态寄存器中位字段的剩余部分 (位 3 到位 6)，如 **Figure 2** 所示。若在 OK ACK 响应条件下设置任何其他位字段，仍需要中止并重启 HSSP 操作。

如果主机编程器尝试提供的 SWDCK 时钟频率高于编程规范中对 SWDCK 指定的最大值，将得到 WAIT ACK 响应。

16.2 位 3 — SWD 读取数据奇偶校验错误

如果在目标器件所收到的数据中发生了奇偶校验错误，主机编程器将置位该位。主机必须中止并重启 HSSP 操作。

16.3 位 4 — 端口获取超时

如果获取目标器件步骤 (第一步: *ProgrammingSteps.c* 文件中的 `DeviceAcquire()`函数部分) 中传输的 SWD 数据包尚未成功完成，该位将置位。如果置位该位，必须中止并重启 HSSP 操作。

该超时错误可能有两种原因: 主机编程器和目标器件之间的硬件连接失败，或者主机编程器未能满足进入目标器件编程模式时序上的要求。

有关进入 PSoC™ 4 编程模式的时序要求的更多信息，请参见**相关文档**一节中所列出的相应器件编程规范文档中“步骤 1. 获取芯片”部分。

16.4 位 5 — SROM 轮询超时错误

PSoC™ 4 的闪存编程是通过使用 SROM API 进行。当 `PollSromStatus()`函数返回一个错误状态时，将置位该位。为了获取状态代码，将使用 1 秒的时间轮询 CPUSS_SYSREQ 寄存器。如果轮询操作超过 1 秒的时间，或者该函数返回的状态代码是 FAILURE，将置位该位。该函数在所有 HSSP 步骤中都被调用，以读取 SROM 系统请求的状态。

如果该位被置位，主机编程器必须调用 *ProgrammingSteps.h* 文件中的 `ReadSromStatus()`函数，以便读取并显示在轮询操作中所返回的状态代码的值。

有关 SROM 状态代码的更多信息，请参见**附录 B**. SROM 请求的状态代码。

16.5 位 6 — 验证失败

该错误位可在多个步骤中置位，验证失败有多种原因，如 **Table 14** 所述。

Table 14 验证错误 — 步骤及原因

错误步骤	错误	错误原因
获取器件 (第一步)	器件 ID 验证的错误	器件返回的 IDCODE 与 Cortex-M0 DAP ID (0x0BB11477) 不匹配。
验证芯片 ID (第二步)	芯片 ID 验证的错误	Hex 文件中的芯片 ID 信息与从目标器件读取的芯片 ID 不匹配。
验证闪存 (第六步)	闪存数据验证的错误	闪存数据与 hex 文件中的数据不匹配。
验证保护设置 (第七步)	闪存保护数据验证的错误	从芯片读取的行保护数据或芯片保护数据与 hex 文件的数据不匹配。
验证校验和 (第九步)	校验和验证错误	目标器件中闪存数据校验和的值与 hex 文件中的校验和不匹配。

从上面内容可见，位 6 可在多种验证错误的情况下置位。根据置位该位的步骤，您可以推断验证失败的原因。例如，如果在“验证芯片 ID”步骤中置位该位，主机编程器应用可确定该错误是由芯片 ID 失配引起的。

16.6 位 7 — 跃变错误

当从芯片读取的和存储在 hex 文件中的芯片保护设置指示错误的跃变时，将置位该位。

欲了解有关保护模式和有效和无效跃变的状态框图的信息，请参见**相关文档**一节中所列出的相应器件编程规范文档中“附录 A: 芯片级别保护”部分。

17 附录 D. HSSP 函数

下列各表列出了 HSSP 固件架构的每一层中所定义的公开函数。这些函数用于实现 HSSP 固件各层之间的通信，如 **Figure 1** 所示。

Table 15 *SWD_PhysicalLayer.h* 中的函数

函数	说明
`SetSwdckHigh()`	将主机 SWDCK 引脚设置为高电平。
`SetSwdioHigh()`	将主机 SWDIO 引脚设置为高电平。
`SetXresHigh()`	将主机 XRES 引脚设置为高电平。
`SetSwdckLow()`	将主机 SWDCK 引脚设置为低电平。
`SetSwdioLow()`	将主机 SWDIO 引脚设置为低电平。
`SetXresLow()`	将主机 XRES 引脚设置为低电平。
`SetSwdckCmosOutput()`	将主机 SWDCK 引脚配置为 CMOS 输出驱动模式。
`SetSwdioCmosOutput()`	将主机 SWDIO 引脚配置为 CMOS 输出驱动模式。
`SetXresCmosOutput()`	将主机 XRES 引脚配置为 CMOS 输出驱动模式。
`SetSwdckHizInput()`	将主机 SWDCK 引脚配置为高阻抗数字输入驱动模式。
`SetSwdioHizInput()`	将主机 SWDIO 引脚配置为高阻抗数字输入驱动模式。
`SetXresHizInput()`	将主机 XRES 引脚配置为高阻抗数字输入驱动模式。
`ReadSwdio()`	返回 SWDIO 输入引脚的当前状态。

Table 16 *SWD_PacketLayer.h* 中的函数

函数	说明
`Swd_WritePacket()`	发送一个 SWD 写入数据包。此函数使用 swd_PacketHeader、swd_PacketAck 和 swd_PacketData[]三个全局变量执行操作。
`Swd_ReadPacket()`	发送一个单一的 SWD 读取数据包。该函数使用 swd_PacketHeader、swd_PacketAck 和 swd_PacketData[]三个全局变量执行操作。该函数用于读取一个特定的地址。
`SwdLineReset()`	通过发送 51 个 SWDCK 时钟周期，同时将 SWDIO 线路设为高电平，可以复位 SWD 线路。用于在编程期间的第 1 步中获取调试访问端口 (DAP)。

Table 17 *SWD_UpperPacketLayer.h* 中的函数

函数	说明
`Read_DAP`	从特定的 DAP 寄存器内读取 32 位数据，并将它写入到 Swd_PacketData[]变量内。该函数使用 Swd_ReadPacket()函数读取数据。
`Write_DAP`	将 32 位数据写入到特定的 DAP 寄存器中。该函数使用 Swd_WritePacket()函数进行写入数据。
`Read_IO`	从 CPU 地址空间中的指定地址读取 32 位数据。使用 Read_DAP()和 Write_DAP()函数执行此函数。若所有 SWD 数据操作都成功 (ACKed)，则返回“true”。
`Write_IO`	将 32 位数据写入到 CPU 地址空间中的指定地址。使用 Write_DAP()函数实现此函数。若所有 SWD 数据操作都成功，则返回“true”。

附录 D. HSSP 函数

Table 18 *Timeout.h* 中的函数

函数	说明
`DelayHundredUs()`	生成了一个时长为 100 µs 的延迟，用于在 XRES 引脚上生成持续 100 µs 的低电平有效脉冲信号。

Table 19 *DataFetch.h* 中的函数

函数	说明
`HEX_ReadSiliconId()`	将 *HexImage.c* 文件中的器件芯片 ID 数据复制到指示的目标阵列内。
`HEX_ReadRowData()`	将 *HexImage.c* 文件中的闪存行数据复制到指示的目标阵列内。此外，还将闪存行编号作为此函数的参数。
`HEX_ReadRowProtectionData()`	将 *HexImage.c* 文件中的闪存行保护数据复制到指示的目标阵列内。保护数据的字节大小也作为此函数的参数。
`HEX_ReadChipProtectionData`	将 *HexImage.c* 文件中的芯片保护数据复制到指示的目标内。
`HEX_ReadChecksumData()`	将 *HexImage.c* 文件中的校验和数据复制到指示的目标内。
`GetFlashRowCount()`	从 *HexImage.c* 文件中返回目标器件的闪存行总数。

Table 20 *ProgrammingSteps.h* 中的函数

函数	说明
`DeviceAcquire()`	进入编程模式，并获取目标器件。
`VerifySiliconId()`	验证目标器件的芯片 ID 是否与 hex 文件中的芯片 ID 相匹配。
`EraseAllFlash()`	删除目标器件的整个闪存，包括闪存保护数据。
`ChecksumPrivileged()`	计算闪存内的特权数据的校验和。
`ProgramFlash()`	对目标器件的闪存进行编程。
`VerifyFlash()`	验证编程到目标器件的闪存数据是否与 hex 文件中的闪存数据相匹配。
`ProgramProtectionSettings()`	将行保护数据和芯片保护数据编程到目标器件内。
`VerifyProtectionSettings()`	验证编程到目标器件的行保护数据和芯片保护数据是否与 hex 文件中的数据相匹配。
`VerifyChecksum()`	验证从目标器件读取的校验和数据是否与 hex 文件的校验和数据相匹配。
`ExitProgrammingMode()`	通过在 XRES 引脚上生成低电平有效脉冲信号退出目标器件编程模式。
`ReadHsspErrorStatus()`	返回 HSSP 操作的错误状态。
`ReadSromStatus()`	在 SROM 轮询超时发生错误的条件下，将返回 CPUSS_SYSARG 状态寄存器的值。

文档修订记录

版本	提交日期	变更说明
**	2014-10-04	本文档版本号为 Rev**，译自英文版 001-84858 Rev*C。
*A	2015-04-10	本文档版本号为 Rev*A，译自英文版 001-84858 Rev*E。
*B	2015-11-05	本文档版本号为 Rev*B，译自英文版 001-84858 Rev*H。
*C	2017-06-13	更新了徽标和版权。
*D	2020-03-31	本文档版本号为 Rev*D，译自英文版 001-84858 Rev*M。
*E	2022-05-27	本文档版本号为 Rev. *E，译自英文版 001-84858 Rev. *N。

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

Edition 2022-05-27
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
001-92086 Rev. *E

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。