# Altera SoC 嵌入式设计套件用户指南

订阅
反馈

ug-1137
2016.02.17

101 Innovation Drive
San Jose, CA 95134
www.altera.com

ALTERA
now part of Intel

# 内容

# SoC 嵌入式设计套件简介

1

2016.02.17

ug-1137 订阅 反馈

Altera®片上系统(SoC)嵌入式设计套件(EDS)提供了对 Altera SoC 器件进行的嵌入式软件开发所需的工具。

Altera SoC EDS 是一种用于 Altera SoC 器件上的嵌入式软件开发的完整工具套件。Altera SoC EDS 包括开发工具、实用程序、运行时软件和应用实例， 实现了 Altera SoC 硬件平台上的固件和应用软件的开发。

## 概述

Altera SoC EDS 使您能够执行针对 Altera SoC 的全部所需的软件开发任务，包括：

- 电路板启动
- 器件驱动程序开发
- 操作系统(OS)移植
- 裸机(bare-metal)应用程序开发和调试
- 基于 OS 和 Linux 的应用程序开发和调试
- 调试运行对称多处理(SMP)的系统
- 调试针对位于器件的 FPGA 部分的软核 IP 的软件

SoC EDS 的主要组件包括：

- ARM® Development Studio 5 (DS-5™) AE (AE) 工具包
- 编译器工具链：
  - Mentor Graphics®提供的裸机 GNU Compiler Collection (GCC)工具
  - 裸机编译器
  - Linaro®提供的 Linux GCC 编译器工具链
- 预构建的 Linux 软件包包括：
  - Linux 内核可执行程序
  - U-boot 映像
  - Bootloader 器件树—A10 nu-boot 使用的器件树 blob
  - Linux 器件树—Linux 使用的器件树 blob
  - 安全数字(SD)卡映像
  - 从 GitHub 库下载 Linux 源代码的脚本(1)
- SoC 硬件库(HWLIB)
- 硬件到软件接口工具：
  - 第二阶段引导加载程序生成器
  - Linux 器件树生成器
- 示例应用程序
- 金牌硬件参考设计(GHRD)包括:
  - FPGA 硬件工程
  - FPGA 硬件 SRAM 目标文件(.sof)
  - 预编译的第二阶段引导程序
    - Arria V 和 Cyclone V 的预加载程序
    - Arria 10 的引导加载程序
- 实现轻松调用所包含工具的嵌入式命令壳
- SD 卡引导实用程序
- Quartus® Prime Programmer and SignalTap II

**注意:** SoC EDS 中包含的 Linux 软件包不是正式版，仅用作示例参考。请使用 Rocketboards 网站上*金牌系统参考设计(GSRD)用户手册*中描述的正式 Linux 版本，或者使用位于 GitHub 库中 Git 树的特定版本。

**注意:** SoC EDS 仅通过与其伴随的 Linux 版本进行测试。更新的 Linux 版本可能不会与此版本的 SoC EDS 完全兼容。

**注意:** SoC EDS 中包含的金牌系统参考设计(GHRD)不是正式版，仅用作示例参考。如要用于开发目的，请使用 Rocketboards 网站上 *GSRD 用户手册*中描述的 GHRD 正式版。

**相关链接**

- **金牌系统参考设计用户手册**
- **GitHub Repository**

## Linux 器件树二进制

作为 SoC EDS 的一部分，有两种不同的 Linux 器件树二进制(DTB)文件版本:

(1) 此脚本下载对应于预构建 Linux 软件包的源代码。

- **prebuilt_images** 文件夹中的版本：**socfpga_cyclone5.dtb** 和 **socfpga_arria10.dtb** 是通用 DTB 文件，不依赖于软核 IP。在 Linux 使用此 DTB 开始运行前，不需要 FPGA 编程和桥接释放。

  此 DTB 文件适合于那些对启动新电路板或者想简化引导流程(直到 Linux 提示)感兴趣的用户。如果所开发和或调试的部分不包括 FPGA，那么最好去除 FPGA 复杂性。
- 硬件设计文件夹的版本：**soc_system.dtb**、**ghrd_5astfd5k3.dtb** 和 **ghrd_10as066n2.dtb** 基于 GHRD 设计，是 GSRD 的一部分。由于 GHRD 不包含软核 IP，因此这些 DTB 文件版本会通知 Linux 加载软核 IP 驱动程序。这样在引导 Linux 前需要编程 FPGA 并释放桥接。

# 硬件和软件开发角色

根据在硬件和软件开发中的不同角色，您需要一组不同的 SoC EDS 工具包。下表列出了一些常用的工程开发角色并显示每种角色通常所需要的工具。

关于这些工具中每种工具的详细信息，请参考 **SoC 嵌入式设计套件**页面。

**表 1-1: 硬件和软件开发角色**

下表列出了工具使用情况，但实际要求取决于特定的工程和组织。

| 工具 | 硬件工程师 | 裸机开发人员 | RTOS 开发人员 | Linux 内核和驱动程序开发人员 | Linux 应用程序开发人员 |
|---|---|---|---|---|---|
| ARM DS-5 Debugging | ✓ | ✓ | ✓ | ✓ | ✓ |
| ARM DS-5 Tracing | | ✓ | ✓ | ✓ | |
| ARM DS-5 Cross Triggering | | ✓ | ✓ | ✓ | |
| Hardware Libraries | | ✓ | ✓ | ✓ | |
| Second Stage Bootloader Generator | ✓ | ✓ | ✓ | ✓ | |
| Flash Programmer | | ✓ | ✓ | ✓ | ✓ |
| Bare-Metal Compiler | ✓ | ✓ | ✓ | ✓ | |
| Linux Compiler | | | | ✓ | ✓ |
| Linux Device Tree Generator | | | | ✓ | |

## 硬件工程师

作为一个硬件工程师，您通常在 Qsys 中设计 FPGA 硬件。您可以使用 ARM DS-5 AE 的调试程序连接到 ARM 内核并测试硬件。DS-5 调试程序的一个方便特性是 soft IP 寄存器的可视性，使用 Cortex Microcontroller Software Interface Standard (CMSIS) System View Description (**.svd**)文件。通过使用此特性，您能够很容易地从 ARM 一侧读取并修改 soft IP 寄存器。

作为一个硬件工程师，您可以对您的硬件配置生成 Preloader。Preloader 是一个软件，根据硬件设计来配置 HPS 组件。

作为一个软件工程师，您也可以执行电路板启动。您可以使用 ARM DS-5 调试程序验证它们是否能够连接到 ARM，并且验证电路板是否工作正常。

这些任务需要 JTAG 调试，只有在订阅版本中才能使能 JTAG 调试。关于详细信息，请参考 *许可* 部分。

**相关链接**

- 许可 (第 3-1 页)
- 硬件 – 软件开发流程 (第 1-5 页)
  关于**.svd** 文件的更多信息，请参考“硬件 – 软件开发流程”部分。

## 裸机和 RTOS 开发人员

作为一个裸机或 RTOS 开发人员，您需要 JTAG 调试和系统的低级别可视性。

使用裸机编译器编译您的代码和 SoC Hardware Library，以方便一致的方式控制硬件。

使用 Flash Programmer 编程目标电路板上的闪存。

这些任务需要 JTAG 调试，只有在订阅版本中才能使能 JTAG 调试。关于详细信息，请参考 *许可* 部分。

**相关链接**

许可 (第 3-1 页)

## Linux 内核和驱动程序开发人员

作为一个 Linux 内核或驱动程序开发人员，您可以使用 RTOS 开发人员所有使用的相同工具，因为您需要对系统的低级别访问和可视性。然而，您必须使用 Linux 编译器，而不是裸机编译器。您可以使用 Linux 器件树生成器(DTG)生成 Linux 器件树。

这些任务需要 JTAG 调试，只有在订阅版本中才能使能 JTAG 调试。

关于详细信息，请参考“许可”部分。

**相关链接**

许可 (第 3-1 页)

## Linux 应用程序开发人员

作为一个 Linux 应用程序开发人员，您要对运行在电路板上的 Linux OS 编写代码。因为 OS 提供所有硬件的驱动程序，所以您不需要通过 JTAG 的低级别可视性。DS-5 提供一个 OS 的详细视图，显示诸如哪些线程在运行，哪些驱动程序被加载等信息。

这些任务不需要 JTAG 调试，您可以在网络版以及订阅版中运行这些任务。关于详细信息，请参考 *许可*部分。

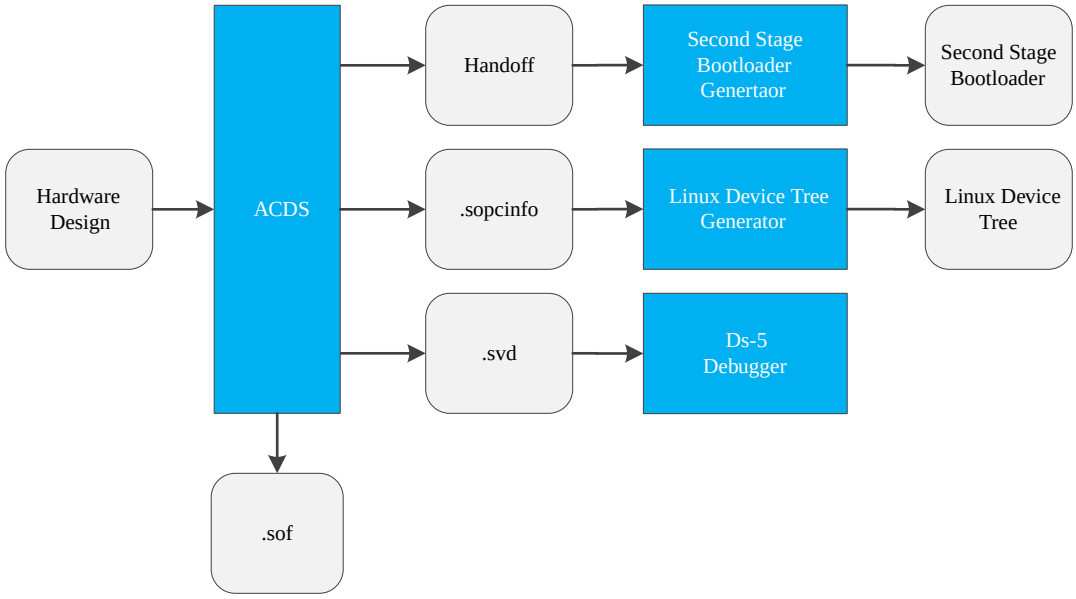
**相关链接**

许可 (第 3-1 页)

# 硬件 – 软件开发流程

Altera 硬件到软件切换(handoff)实用程序使硬件和软件团队能够独立地工作，并遵循他们各自熟悉的设计流程。

**图 1-1: Altera 硬件到软件切换**




当编译硬件工程时，将会创建下面的切换文件：

- **Handoff** 文件夹 – 包含关于如何配置 HPS 的信息，例如：要使能的外设，管脚 MUXing 和 IOCSR 设置，存储器参数等
- **.svd** 文件 – 包括 HPS 寄存器的描述和器件的 FPGA 部分中实现的 FPGA 一侧上的软核 IP 寄存器的描述
- **.sopcinfo** 文件 – 包含整个系统的描述

第二阶段引导加载生成器使用切换(handoff)文件夹来创建预加载器。

关于切换文件夹(handoff folder)的更多信息，请参考 *BSP 生成流程*章节。

**.svd** 文件包含 HPS 外设寄存器的寄存器描述和 SoC 的 FPGA 部分中软核 IP 组件的寄存器描述。ARM DS-5 Debugger 使用此文件，使用户能够检查并修改这些寄存器。

SOPC 信息(**.sopcinfo**)文件，包含整个系统的描述，Linux 器件树生成器使用此文件来创建由 Linux kernel 使用的器件树。

关于更多信息，请参考"Linux 器件树生成器"章节。

**注意:** 生成的软核 IP 寄存器描述不并适用于所有的软核 IP 内核。

**相关链接**

- BSP 生成流程 (第 7-2 页)

- Linux 器件树生成器
- SoC 嵌入式设计套件下载页面

## SoC EDS 简介文档修订历史

| 日期 | 版本 | Changes |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 维护版本。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# 安装 Altera SoC 嵌入式设计套件

# 2

2016.02.17

ug-1137



您必须安装 Altera SoC 嵌入式设计套件(EDS)和 ARM DS-5 AE 才能在 Altera SoC 硬件平台上运行 SoC EDS。

## 安装文件夹

默认的 SoC EDS 安装文件夹：

- *<SoC EDS 安装目录>*
  - **c:\altera\15.1\embedded** on Windows
  - **~/altera/15.1/embedded** on Linux

Quartus Prime Programmer 的默认安装目录是：

- *<Quartus installation directory>*
  - **c:\altera\15.1\qprogrammer** on Windows
  - **~/altera/15.1/qprogrammer** on Linux

**注意:** 安装目录定义如下：

- *<Altera 安装目录>*表示 Altera 工具的安装位置。
- *<SoC EDS 安装目录>*表示 SoC EDS 的安装位置。

## 安装 SoC EDS

按照下面步骤在基于 Windows 的系统中安装 SoC EDS 工具套件：

1. 从 Altera 网站的 *SoC 嵌入式设计套件下载中心*页面下载最新的安装程序。
2. 运行安装程序(installer)，打开 **Installing SoC Embedded Design Suite (EDS)**对话框，点击 **Next**，开始 **Setup Wizard**。
3. 接受许可协议并点击 **Next**。
4. 接受默认的安装目录，或者使用其他安装目录，然后点击 **Next**。

   **注意:** 如果您之前已经安装了 Quartus Prime 软件，那么接受默认的*<SoC EDS installation directory>*，以使 Quartus Prime 软件与 SoC EDS Tool Suite 一起运行。

5. 选择 **All** 安装全部组件，然后点击 **Next**。安装程序显示安装汇总。
6. 点击 **Next** 开始安装进程。安装程序显示组件安装进程的一个单独对话框。
7. 安装完成后，开启 **Launch DS-5 Installation**，开始 ARM DS-5 的安装，然后点击 **Finish**。

**注意:** 在一些基于 Linux 的机器上，您可以安装与基于 Windows 的 setup GUI 类似的 setup GUI 的 SoC EDS。由于各种 Linux 发行版和软件包要求，不是所有的 Linux 机器都能够使用 setup GUI。如果不能使用 GUI，那么可以使用等同的命令行程序。从 Altera 网站上的 *SoC Embedded Design Suite Download Center* 页面下载 Linux 安装程序。

**相关链接**

**SoC Embedded Design Suite Download Center**

# 安装 ARM DS-5 Altera 版工具包

### 开始之前

在 SoC EDS 安装进程的最后一步，开始 ARM DS-5 AE Toolkit 的安装。

**注意:** 要确保访问以太网的设置要正确。

1. 当出现 **Welcome** 消息时，点击 **Next**。
2. 接受许可协议并点击 **Next**。
3. 接受默认的安装路径，以确保 SoC EDS 与 ARM DS-5 AE 之间的正确互操作性，然后点击 **Next**。
4. 点击 **Install** 开始安装进程。出现进度条。
5. 当出现驱动程序安装窗口，点击 **Next**。
6. 接受驱动程序安装并点击 **Install**。
7. 安装成功后，点击 **Finish**。ARM DS-5 AE 安装完成。
8. 点击 **Finish**。

# 安装 Altera SoC 嵌入式设计套件文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 安装路径更新到 15.1。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# 3 许可

2016.02.17

ug-1137 订阅 反馈

SoC EDS 有三种不同的许可选项：

- 订购版
- 免费网络版
- 30 天评估订购版

唯一受所选许可选项影响的工具是 ARM DS-5 AE。所有其他工具在所有许可选项中均提供相同等级的功能特性；例如，无论使用哪种许可选项，第二阶段引导加载生成器和裸机编译器都提供相同的功能特性。

许可选项之间主要区别取决于使能的类型和调试方案：

| 许可选项 | 使能的调试方案 |
|---|---|
| 网络版 | • 通过以太网的 Linux 应用程序调试 |
| 订购版<br>30 天评估订购版 | • 基于 JTAG 的裸机调试<br>• 基于 JTAG 的 Linux 内核和驱动程序调试<br>• 通过以太网的 Linux 应用程序调试 |

## 获得许可

根据不同的许可选项，必须按照每个选项的具体步骤来获得许可。

**订购版**-如果您已经购买了 SoC EDS 订购版，那么就已经收到了 ARM 许可序列号。 这是一个由两条短线连接的 15 位字母数字串。您将需要在 DS-5 中使用此序列号来激活您的许可，如*激活许可*部分所示。

**免费网络版**-对于免费的 SoC EDS 网络版，您将能够永久使用 DS-5 通过 Ethernet 连接进行 Linux 应用程序的调试。请从 Altera 网站上的 SoC Embedded Design Suite 下载页面获得 ARM 许可激活码，然后在 DS-5 中激活您的许可，如*激活许可*部分所示。

**30 天评估订购版**-如果您想要评估 SoC EDS 订购版，那么您可以从 Altera 网站上的 SoC Embedded Design Suite 下载页面获得一个 30 天评估激活码，然后在 DS-5 中激活您的许可，如*激活许可*部分所示。

**相关链接**

- SoC EDS 下载页面

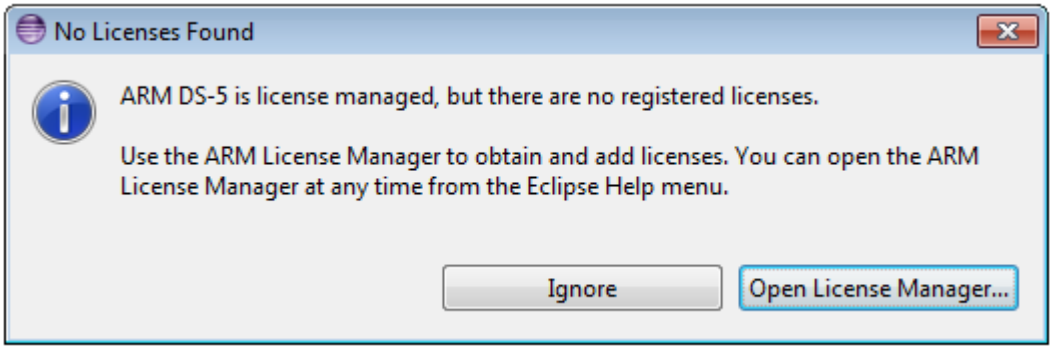
# 激活许可

这一部分介绍在"获得许可"章节中提及的使用许可序列号或激活码在 ARM DS-5 AE 中激活许可的步骤。

**注意:** 需要一个活动用户账户来激活 DS-5 AE 许可。如果您没有活动用户账户，那么可以在 ARM 网站的 ARM *Self-Service* 页面上创建一个。

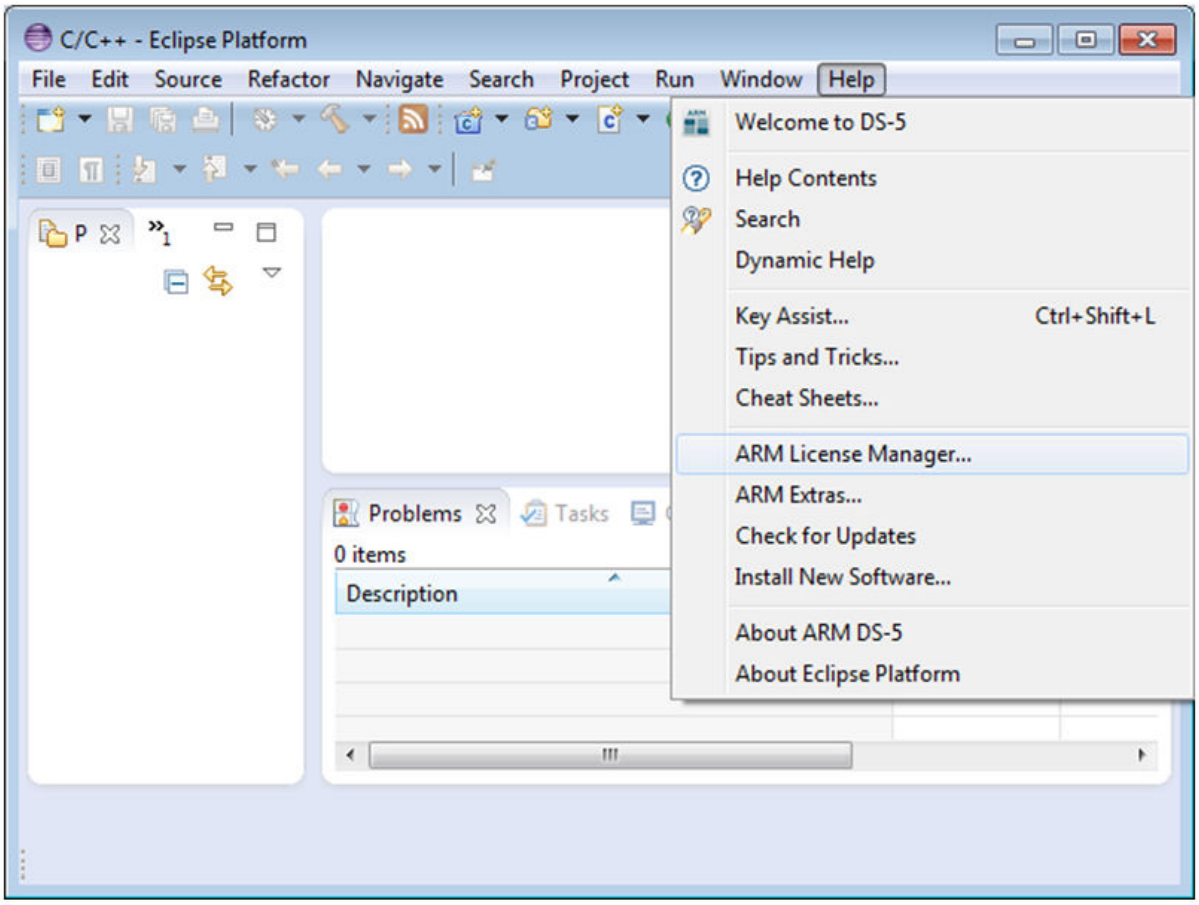
1. ARM DS-5 第一次运行时会提示您需要一个许可。点击 **Open License Manager** 按钮。

**图 3-1: 没有找到许可(No License Found)**



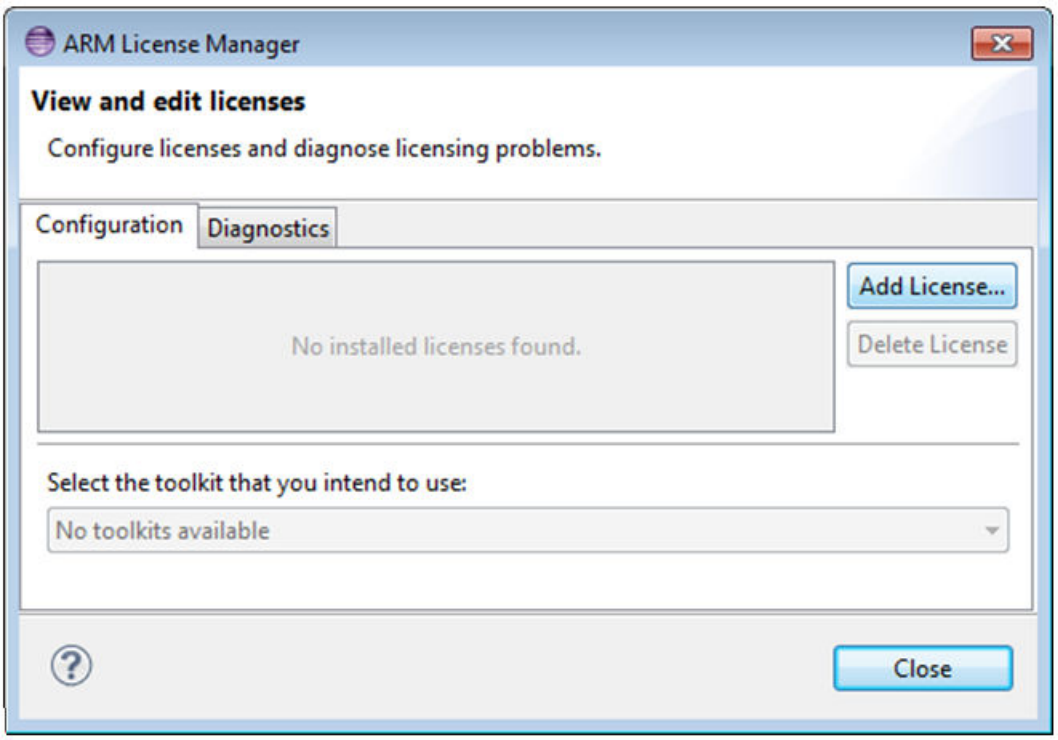

2. 如果任何时候需要修改许可，那么选择 **Help** > **ARM License Manager**，打开 **License Manager**。

**图 3-2: 访问 ARM License Manager**



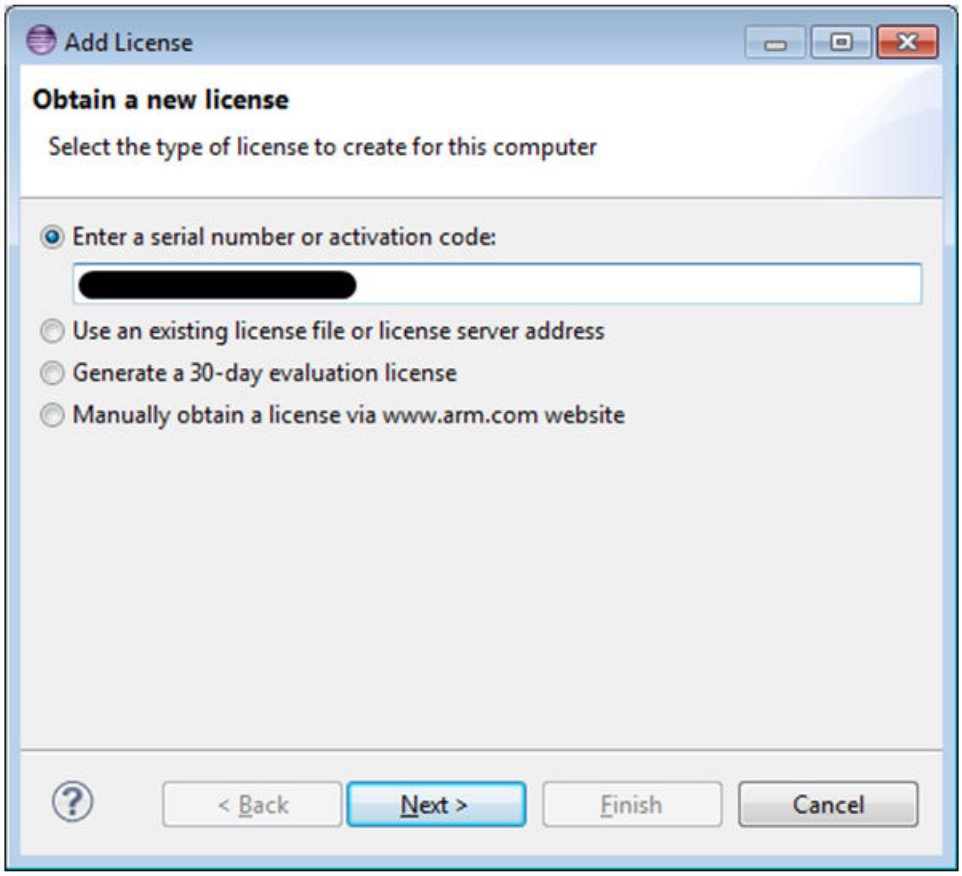

3. **License Manager - View and edit licenses** 对话框打开并显示许可不可用。点击 **Add License** 按钮。

**图 3-3: ARM License Manager**



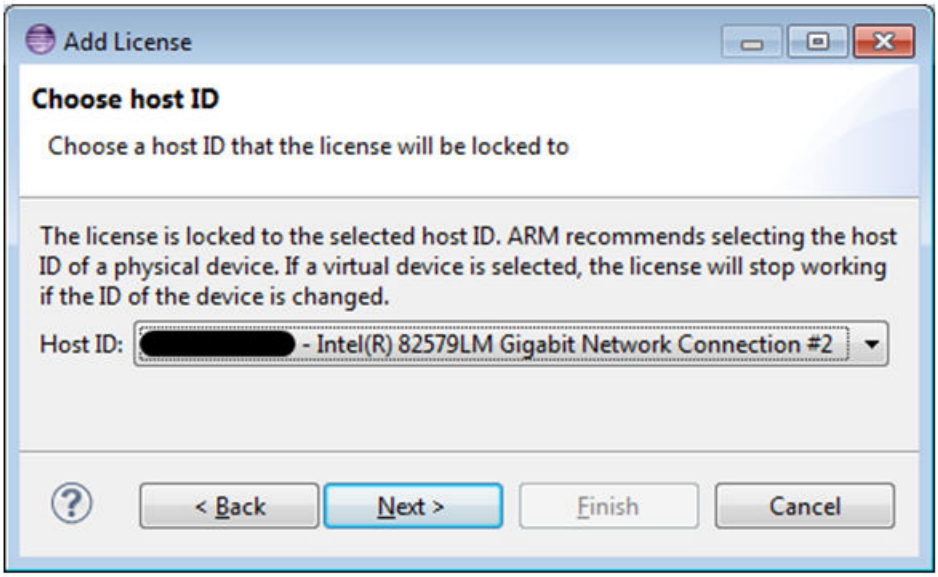

4. 在 **Add License - Obtain a new licenses** 对话框中，选择要输入的许可的类型。在此示例中，选择 radio 按钮，**“Enter a serial number or activation code to obtain a license”**输入列出的选择。完成后点击 **Enter**。
   a. **Subscription Edition** 的 ARM 许可号码。
   b. **Web Edition** 和 **30-Day Evaluation** 的 ARM 许可激活码。

**图 3-4: Add License - Obtain a New License**



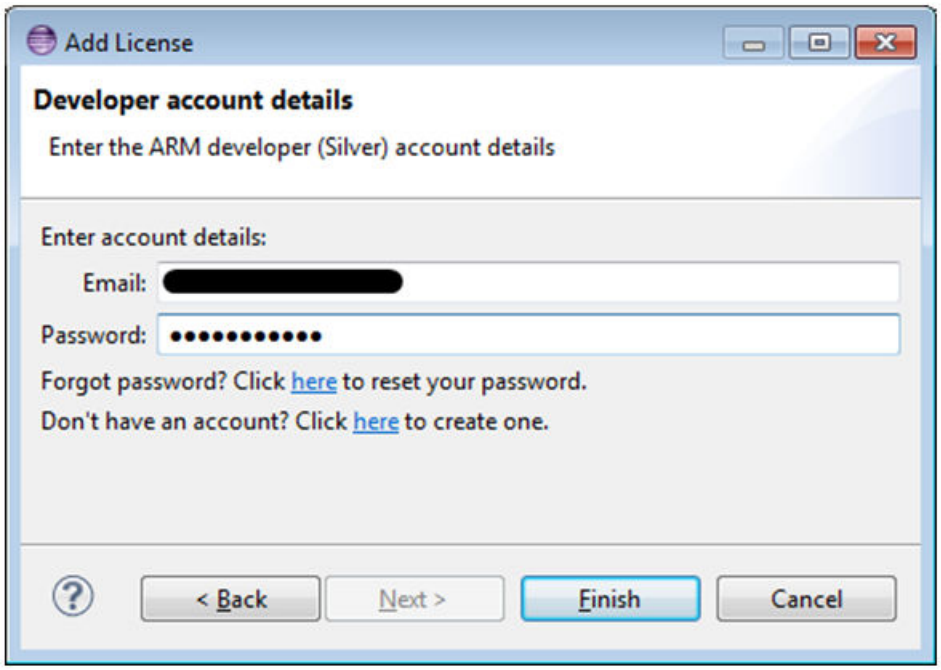

5. 单击 **Next**。
6. 在 **Add License - Choose Host ID** 对话框中，选择要绑定到许可的 Host ID (Network Adapter MAC 地址)。如果有一个以上选项，那么选择您想要锁定许可的那个选项，然后点击 **Next**。

**图 3-5: Add License - Choose host ID**




7. 在 **Add License - Developer account details** 对话框，输入 ARM 开发人员(Silver)账户。如果没有账户，那么可以通过提供的链接轻松创建。输入账户信息后，点击 **Finish**。

**图 3-6: Add License - Developer Account Details**



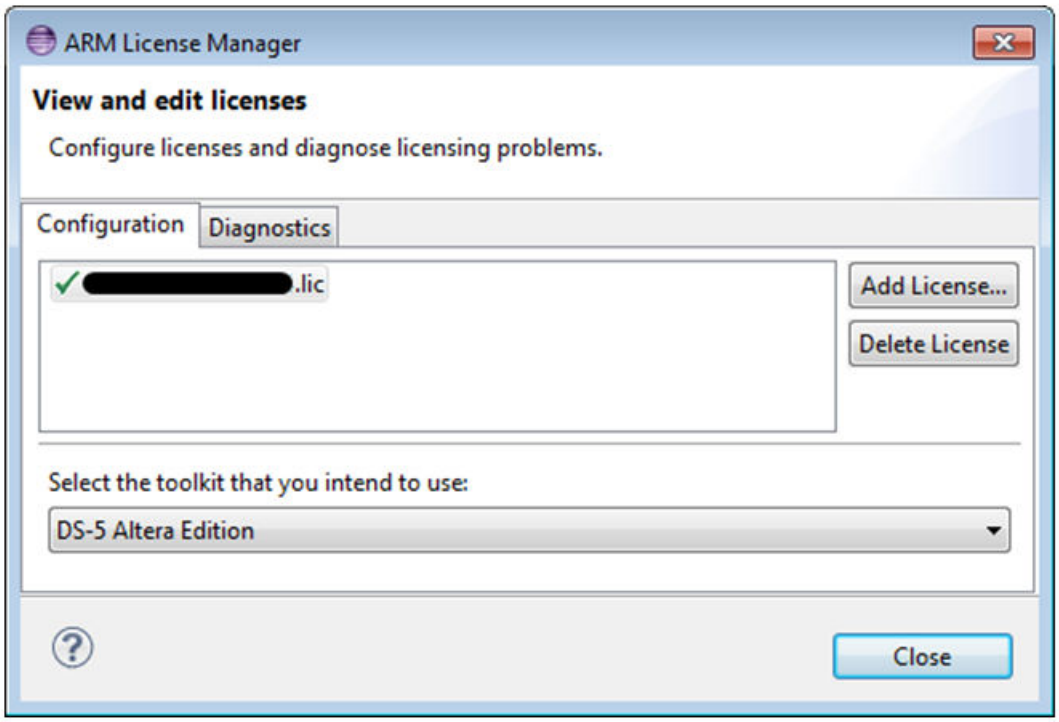

**注意:** License Manager 需要能够连接到 Internet 来激活许可。如果没有 Internet 连接，那么您将需要写下您的 Ethernet MAC 地址，从 ARM 网站的 *ARM Self-Service* 页面直接生成许可， 然后选择 License Manger 中的"**Already have a license**"选项。

**注意:** 只有订购版(具有一个相关联的许可号)能够以这种方式激活。网络版和评估版都是基于激活码的，这些激活码不能在 ARM 网站上的 *ARM Self-Service* 页面上使用，它们需要在 License Manager 中直接输入，这就意味着必须要有 Internet 连接才能获得许可。

ARM License Manager 使用 Eclipse 设置来连接到 Internet。默认的 Eclipse 设置使用系统范围配置访问 Internet。在 License Manager 不能连接到 Internet 的情况下，您可以通过 **Window** > **Preferences** > **General** > **Network Connections** 尝试修改 Proxy 设置。要确保配置并使能"HTTPS"代理项。

8. 片刻之后，`ARM DS-5` 将激活许可并显示在 **License Manager** 中。点击 **Close**。

**图 3-7: ARM License Manager**




**相关链接**

- ARM 网站
- 获得许可 (第 3-1 页)

# SoC EDS 许可文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 维护版本。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# 嵌入式命令壳 4

2016.02.17

ug-1137 订阅 反馈

嵌入式命令壳的目的是提供给您一个调用 SoC EDS 工具的选项，无需通过完整路径对 SoC EDS 工具进行验证就能调用。诸如‘eclipse’，‘bsp-editor’或‘arm-altera-eabi-gcc’的命令可以直接执行。

在 Windows 系统中，通过运行**<SoC EDS installation directory>\Embedded_Command_Shell.bat** 启动命令壳。

在 Linux 系统中，从 **Start** 菜单启动嵌入式命令壳，或者通过运行*<SoC EDS installation directory>* **/embedded_command_shell.sh** 启动嵌入式命令壳。

## 嵌入式命令壳文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.0217 | 维护版本。 |
| 2015 年 8 月 | 2015.0806 | 增添了 Arria 10 支持。 |

# 入门指南 5

2016.02.17

ug-1137 订阅 反馈

入门指南章节提供了如何访问完整的 Getting Started 指令的相关指导，包括：

- Board Setup(电路板设置)
- Running Linux(运行 Linux)
- Running the Tools(运行工具)
- Second Stage Bootloader(第二阶段引导加载器)
- Baremetal Debugging(裸机调试)
- Hardware Libraries (HWLibs)(硬件库)
- Peripheral Register Visibility(外设寄存器可视性)
- Baremetal Project Management(裸机工程管理)
- Linux Application Debugging(Linux 应用程序调试)
- Linux Kernel and Driver Debugging(Linux 内核和驱动程序调试)
- Tracing(跟踪)

**相关链接**

**SoCEDSGettingStarted Wiki Page**
提供关于如何访问“入门指南”的更多信息。

## 入门指南文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.0217 | 维护版本。 |
| 2015 年 8 月 | 2015.08.06 | 删除了此部分中的内容，并移到 Altera Wiki 中。 |

**相关链接**

**SoCEDSGettingStarted**
关于入门指南的详细信息，请参考 Altera Wiki 上的 SoCEDSGettingStarted 页面。

# ARM DS-5 AE 6

2016.02.17

ug-1137 订阅 反馈

ARM Development Studio 5 Toolkit AE (ARM DS-5 AE)是 Altera 提供的特定器件的独家产品。

ARM DS-5 AE 是一个功能强大的基于 Eclipse 的综合集成开发环境（IDE）。一些最重要的特性如下：

- 文件编辑，支持语法高亮和源代码索引
- 构建支持，基于 makefiles
- 裸机调试
- Linux 应用程序调试
- Linux 内核和驱动程序调试
- 多核调试
- 对 HPS 外设寄存器的访问
- 对 FPGA 软核 IP 外设寄存器的访问
- 通过 Program Trace Macrocells (PTM)跟踪程序的执行
- 通过 System Trace Macrocells (STM)跟踪系统事件
- HPS 和 FPGA 之间的交叉触发
- 连接到使用 Altera USB Blaster™ II 的目标器件

ARM DS-5 AE 是一款具有很多特性及选项的复杂工具。本章仅对那些最常用的特性和选项进行描述，并提供入门场景帮助您快速入门。

您可以通过 **Help** > **Help Contents** > **ARM DS-5 Documentation** 或者在线从 Eclipse 访问 ARM DS-5 AE 参考资料。

**相关链接**

**在线 ARM DS-5 文档**
您可以在 ARM 网站的文档页面上在线访问 ARM DS-5 AE 参考材料。

## 启动 Eclipse ARM DS-5 AE

ARM DS-5 AE 必须从嵌入式命令壳启动。

Eclipse 需要从嵌入式命令壳启动，这样所有的工具都添加到搜索路径，能够直接从 makefile 使用这些工具，无需完整路径。

要启动 ARM DS-5 AE 使用的 Eclipse IDE，您必须在命令行输入 **eclipse &**。

# 裸机工程管理

ARM DS-5 AE 使用两种不同的方法对逻辑工程进行方便的工程管理：

- 使用 Makefile
- 使用 ARM DS-5 AE 图形界面

因为 Makefile 支持从脚本执行工程编译的选项，因此一些用户更倾向于使用 Makefile。其他用户则更倾向于使用 GUI 来管理工程， 可管理 GCC 以及 ARM Compiler 裸机工程。

| 方法 | 优势 | 编译器工具链支持 |
|---|---|---|
| Makefile | 脚本化编译 | GCC |
| ARM DS-5 AE 图形界面 | 多工具链支持 | GCC, ARM Compiler |

## 使用 Makefile 的裸机工程管理

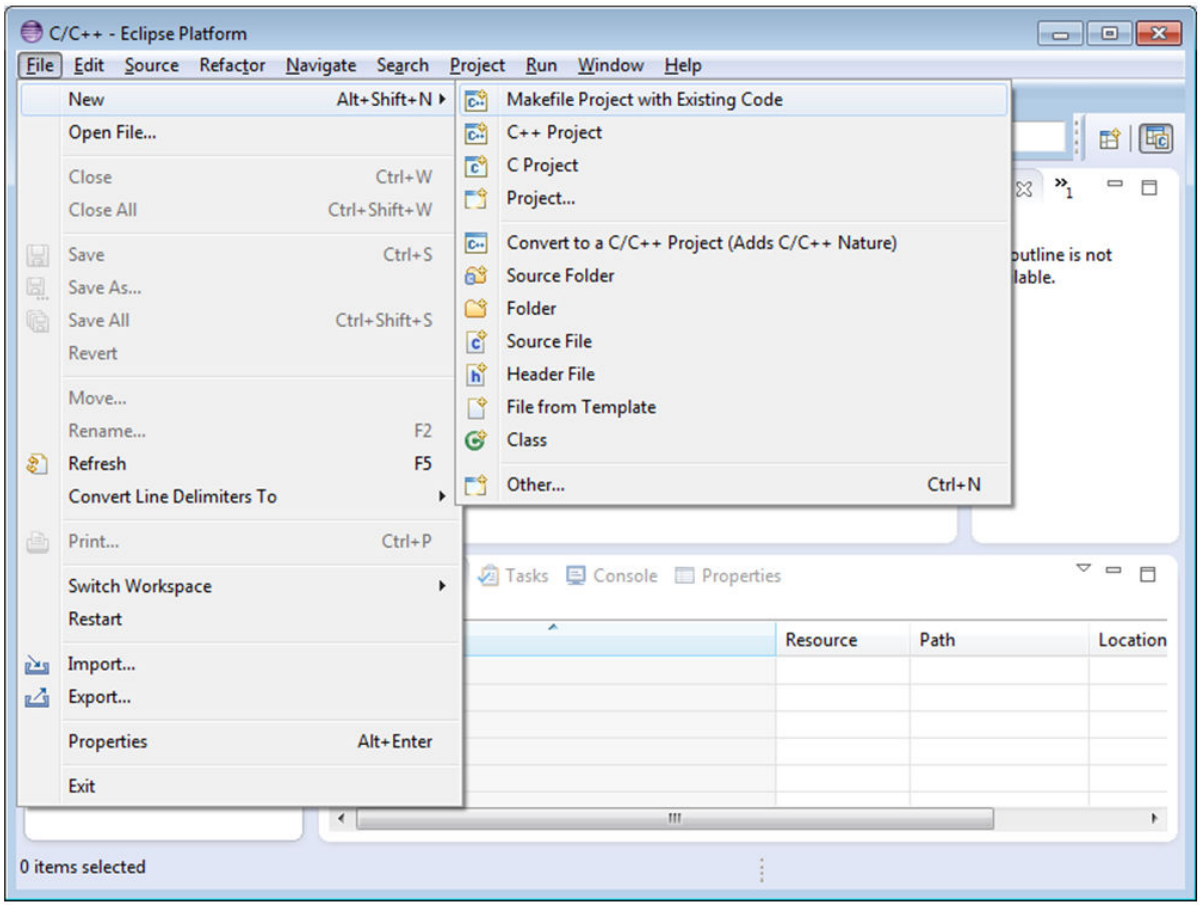
ARM DS-5 AE 通过使用 makefile 使工程管理变得更容易。SoC EDS 提供的样本工程使用 makefile 来管理构建进程。

**注意:** 此选项仅指 DS-5 具体方面。如果您对定义和使用 makefile 不是很熟悉，那么请使用 ARM DS-5 AE GUI 选项，此选项在接下来的部分有详细介绍。

为使 ARM DS-5 AE 能够管理基于 makefile 的工程，需要按照下面步骤创建一个工程：

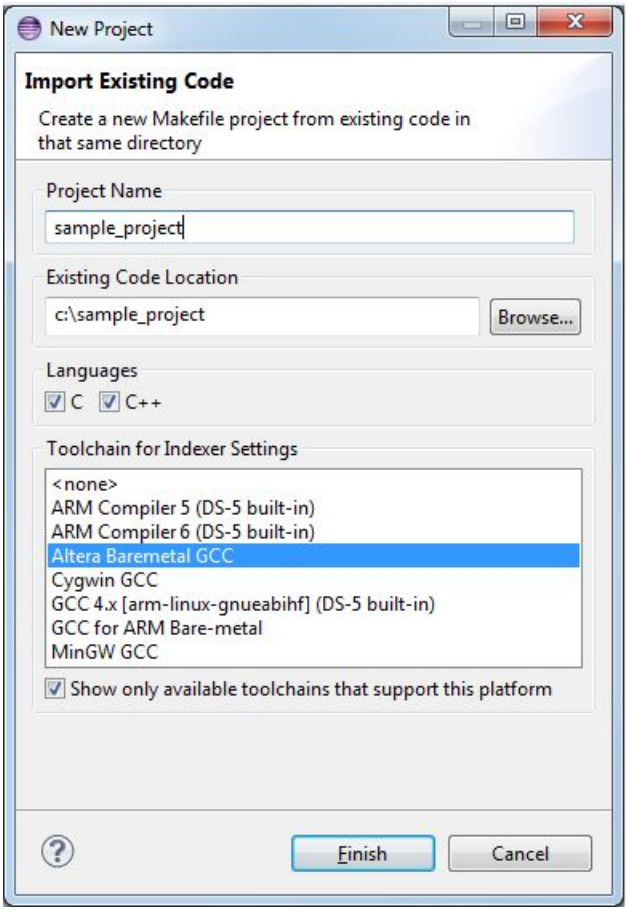
1. 在磁盘上创建一个文件夹。
2. 通过选择 **File > New > Makefile Project with Existing Code** 创建工程。

**图 6-1: 使用现有的代码创建一个工程**



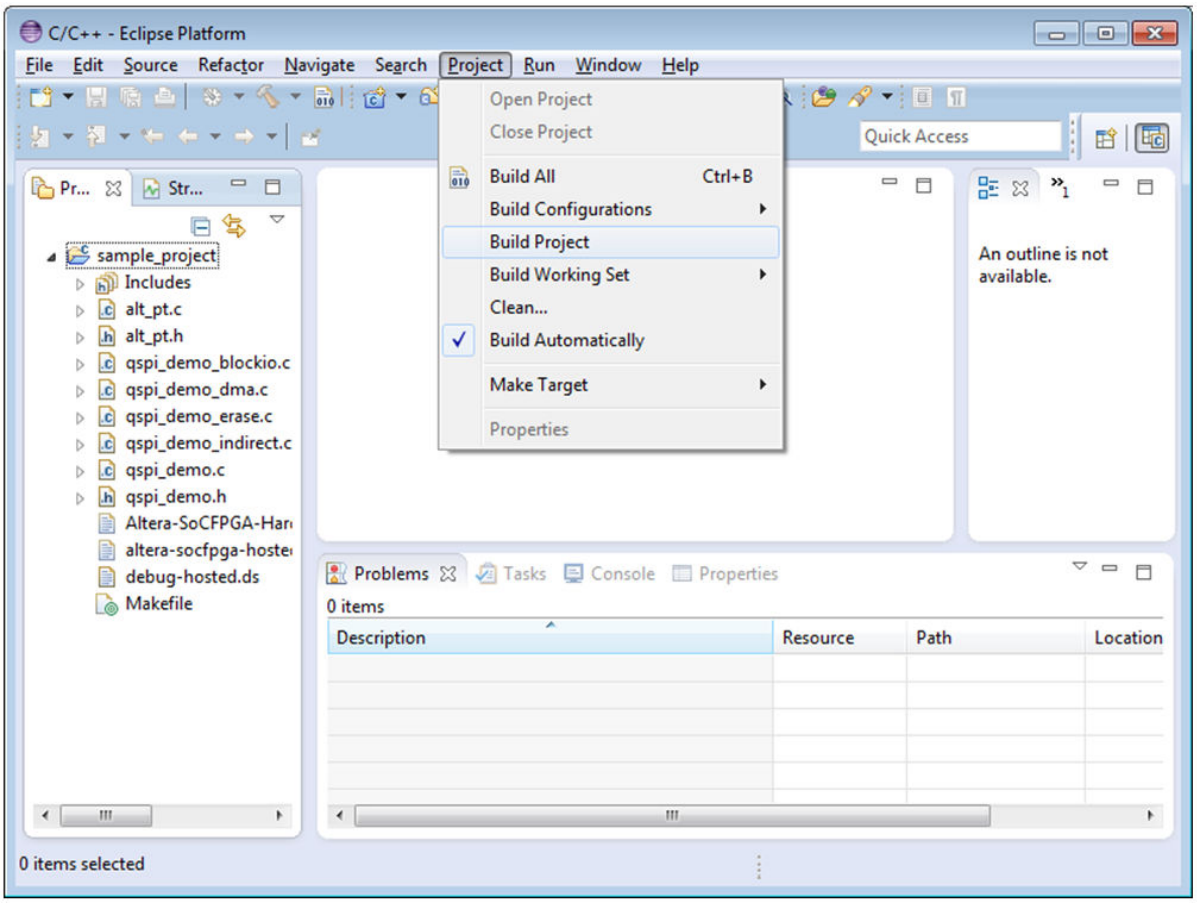

3. 在 **Existing Code Location** 编辑框中输入文件夹名称，然后点击 **Finish**。

**图 6-2: 导入现有代码**




4. 在此文件夹中创建一个 Makefile，并定义编译代码所要求的规则。确保有 **all** 和 **clean** 目标。现在 ARM DS-5 AE 使从 IDE 调用构建进程成为可能，并使您能够构建您的工程，如下图所示：

**图 6-3: Eclipse IDE - Build Process Invoked - 在 ARM DS-5 AE 中构建一个软件工程**




如果编译工具发出错误，那么 ARM DS-5 AE 解析并格式化这些错误，并在 Problems 视窗中显示这些错误。

5. 在 ARM DS-5 AE 中构建一个软件工程。

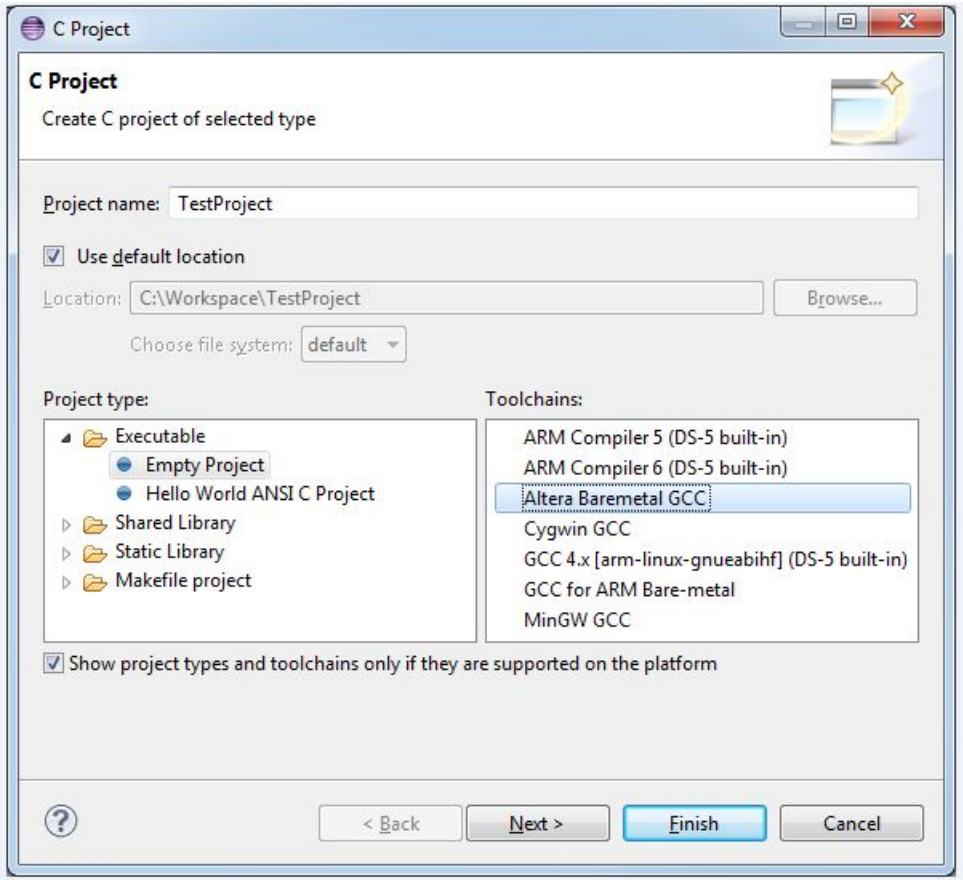
## 基于 GCC 的裸机工程管理

此部分介绍如何使用裸机工具链插件在 GUI 环境中管理基于 GCC 的工程。

### 创建工程

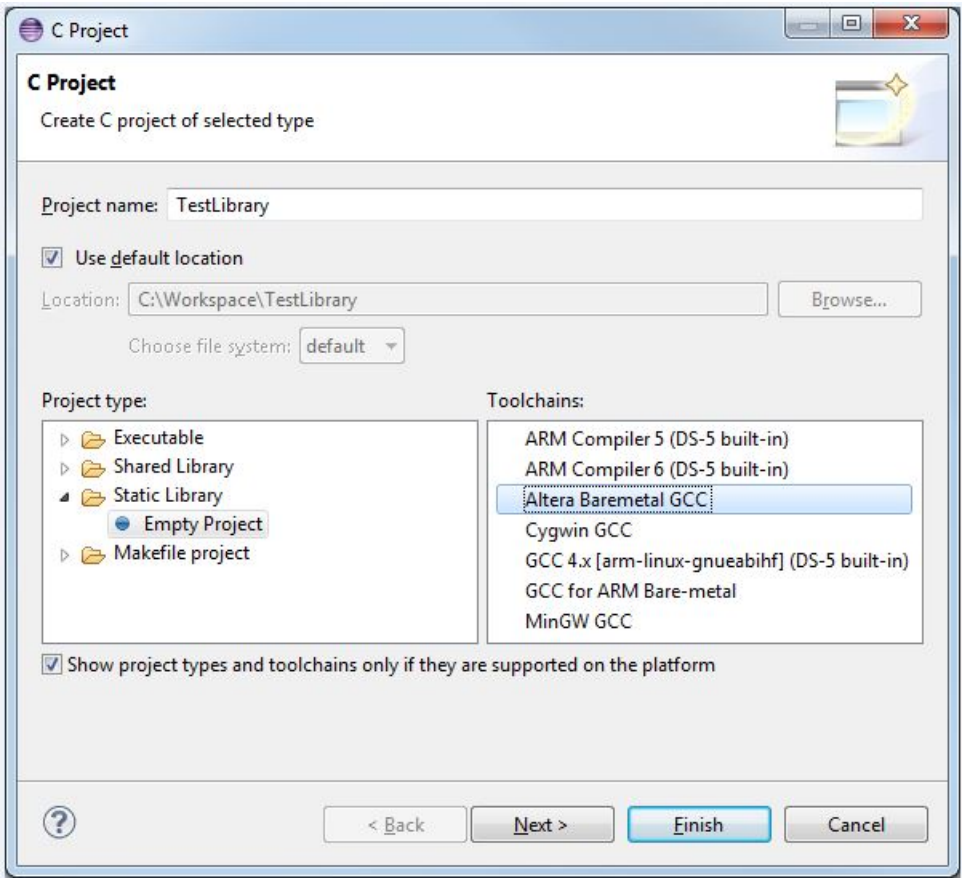
1. 开始 Eclipse。
2. 去到 **File** > **New C Project**。
3. 决定创建一个 **Executable** 空工程还是一个 **Bare-metal library** 空工程。
   a. **Bare-metal executable**
   **Project Type** 选作 **Executable** > **Empty Project**，**Toolchain** 选作 **Altera Baremetal GCC**，然后点击 **Finish**。

**图 6-4: 裸机可执行工程类型**




b. **Static Library（静态库）**
选择 **Bare-metal Library** > **Empty Project** 并点击 Finish。

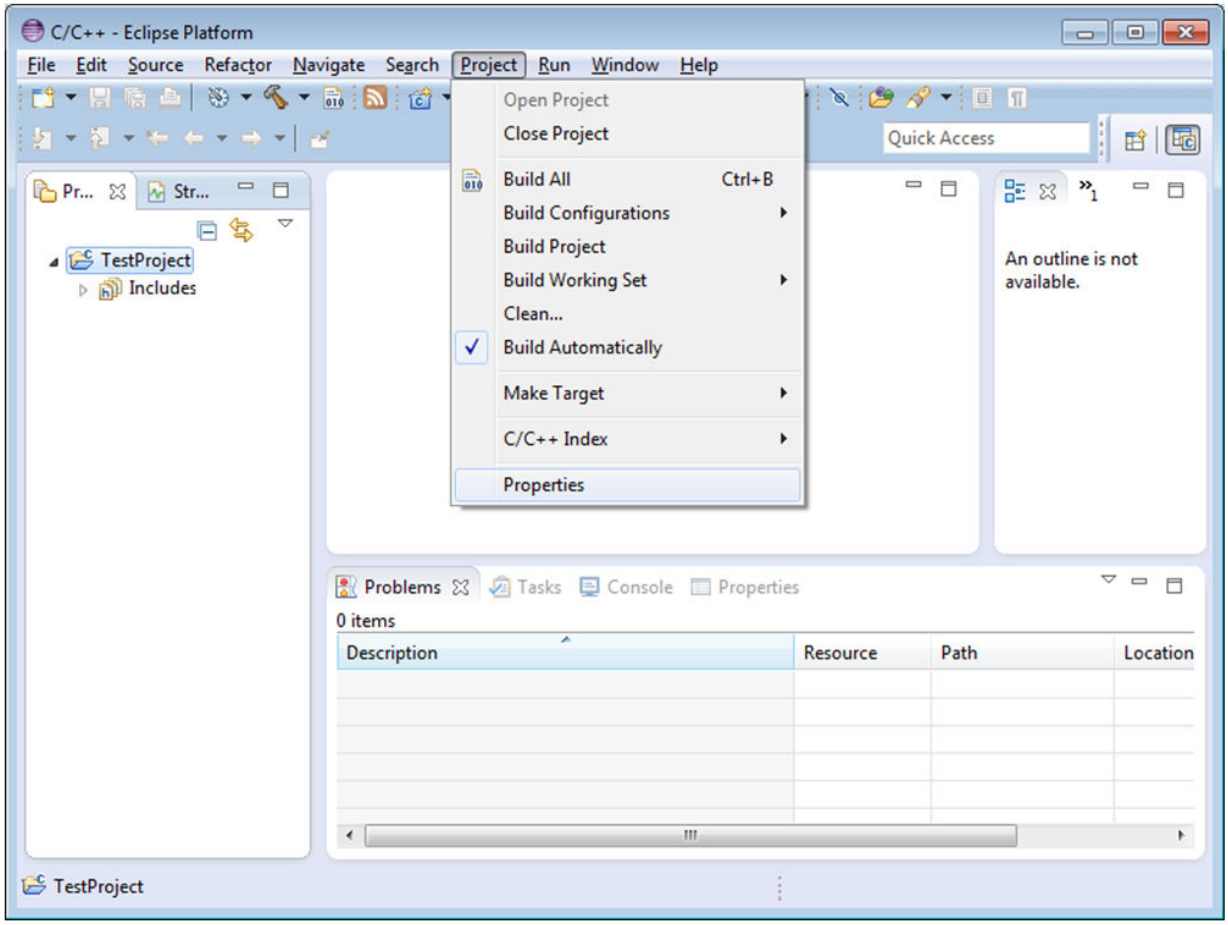
**图 6-5: 裸机库工程类型**




## Build 设置

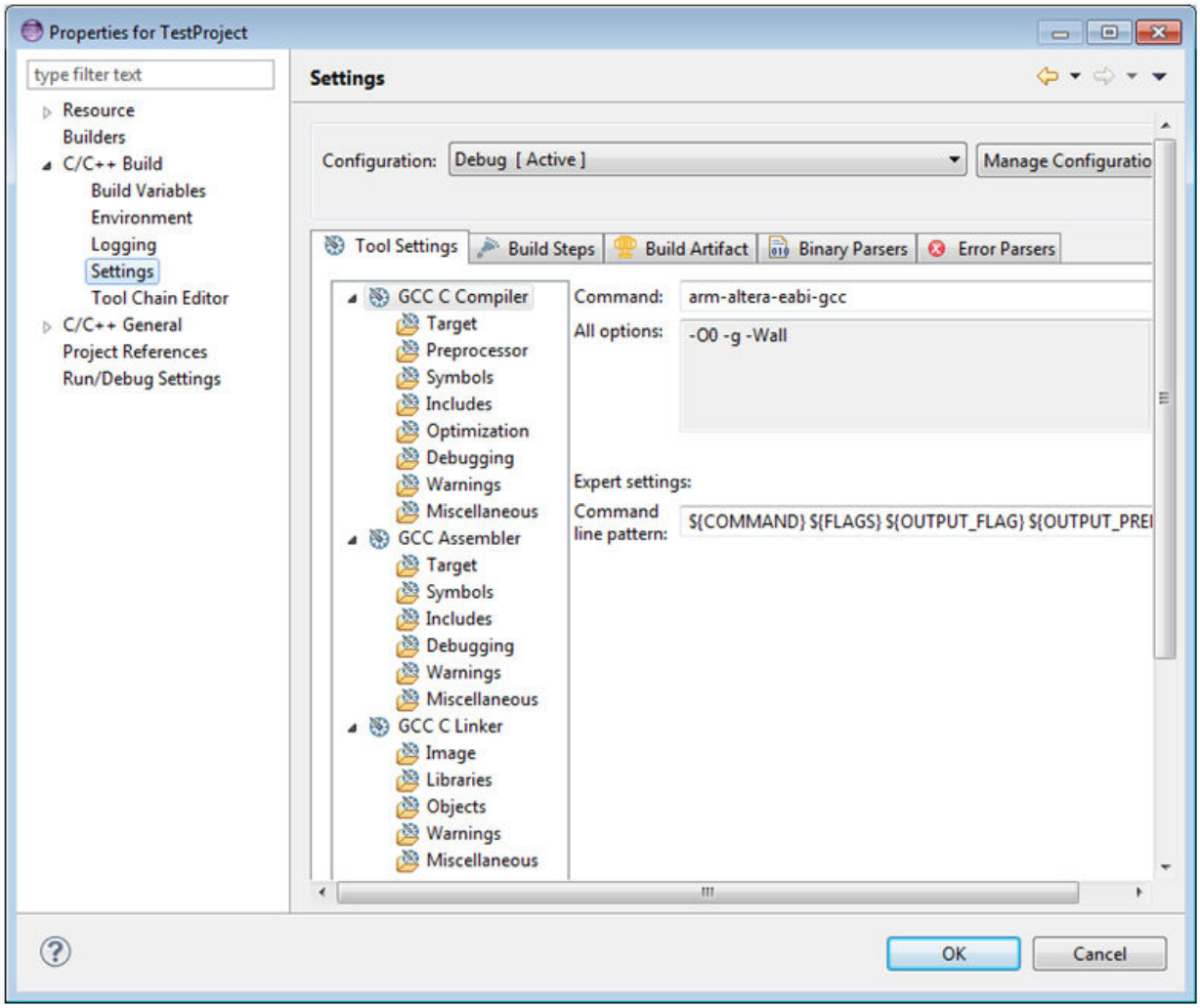
工程一旦创建，就可以通过 **Project** > **Properties** 访问此工程的属性。

**图 6-6: 工程属性**




然后，在 **Project Properites** 窗口中可以通过选择 **C/C++ Build** > **Settings** 来访问 **Compilation** 设置。

**图 6-7: 工程设置**




**Build Settings** 包括所有工具的详细设置：

- Compiler
- Assembler
- Linker

此文档中的*入门指南*章节中包含一个如何在 Altera SoC 开发板上创建一个工程，编译并运行的完整指南的链接。

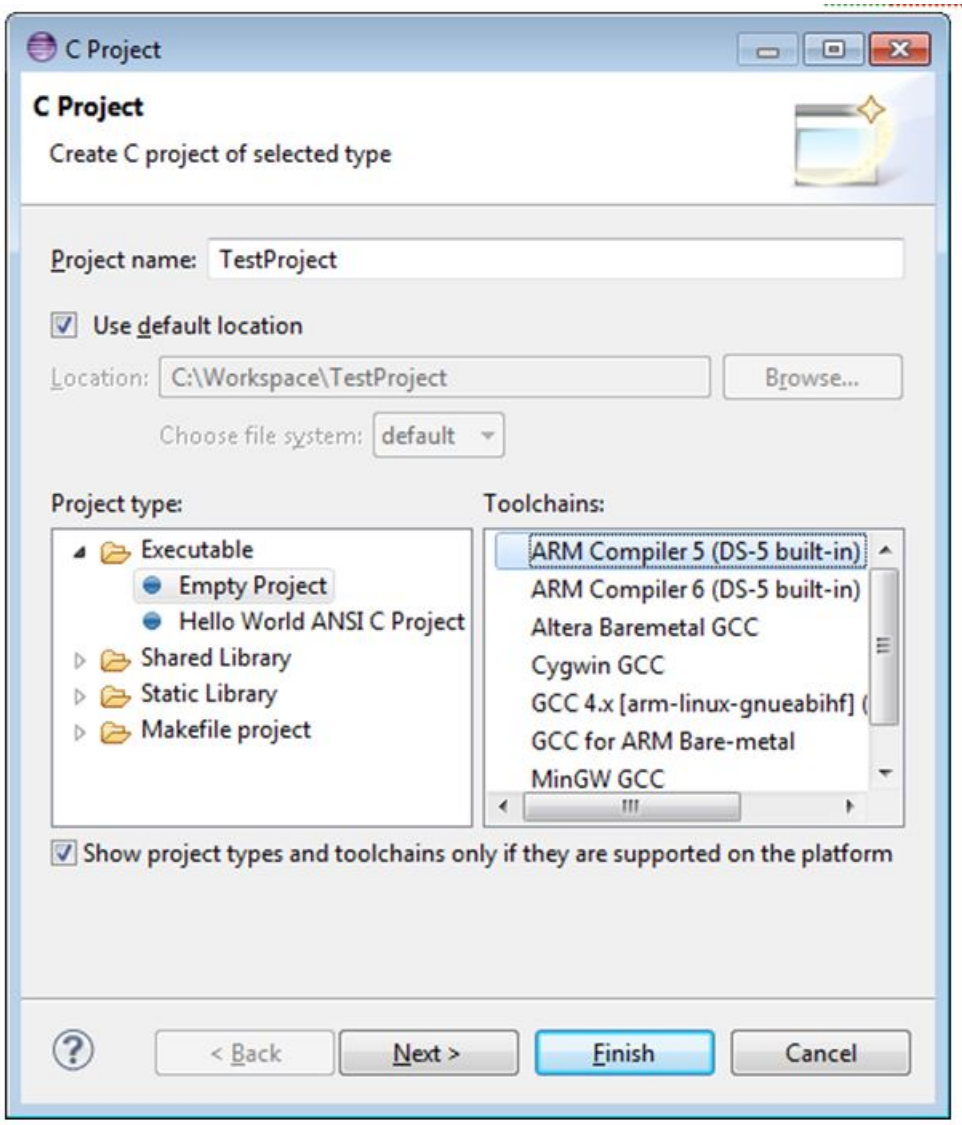
## ARM 编译器裸机工程管理

此部分介绍了如何使用 ARM DS-5 在 GUI 环境中管理 ARM compiler 工程。

## 创建一个工程

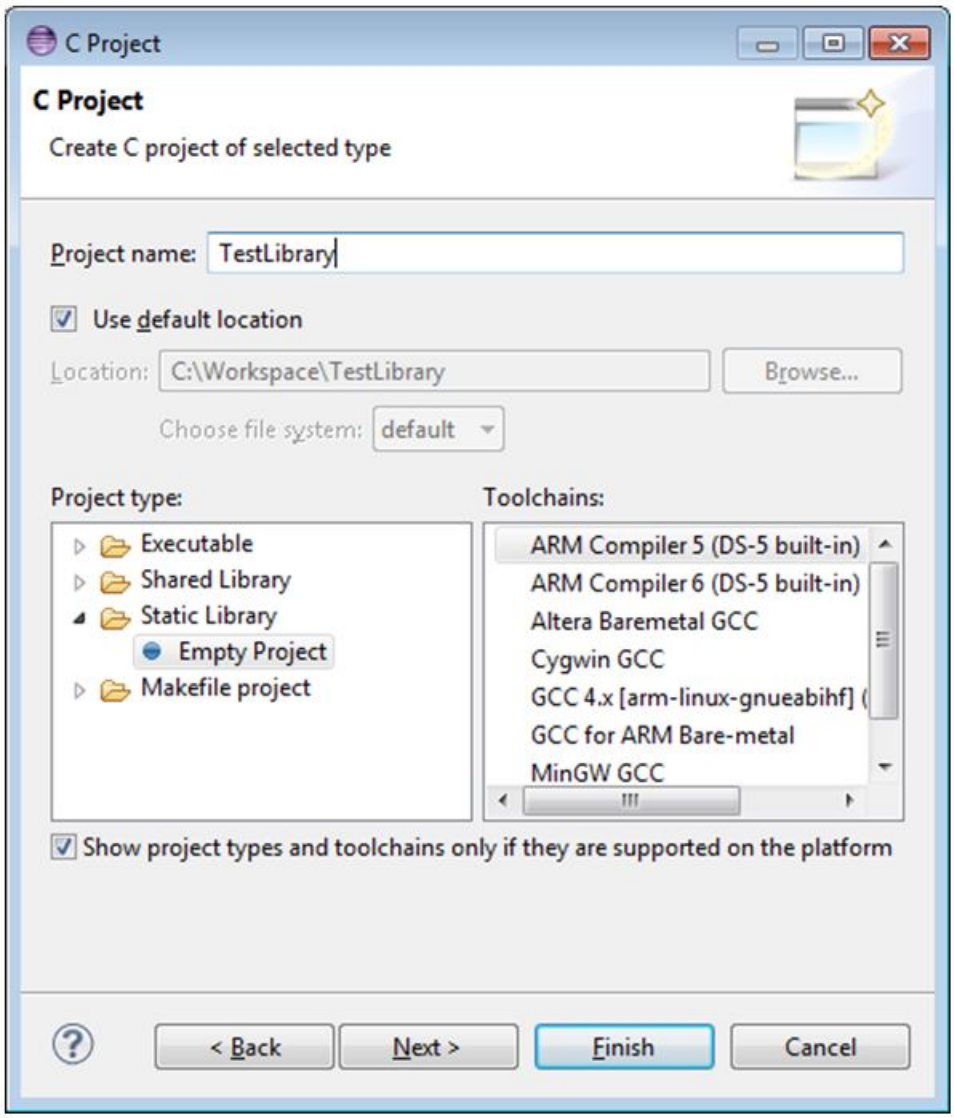
1. 开始 Eclipse。
2. 去到 **File** > **New C Project**。
3. 选择下面其中一个选项：
   a. "Project Type"选作 **Executable** > **Empty Project**，"Toolchains"选作 **ARM Compiler 5**。点击 **Finish**。

**图 6-8: 创建一个空的 ARM Compiler 裸机可执行工程**




   b. 选择 **Static Library** > **Empty Project**，然后"Toolchains"选作 **ARM Compiler 5**。点击 **Finish**。

**图 6-9: 创建一个空的 ARM Compiler 裸机库工程**

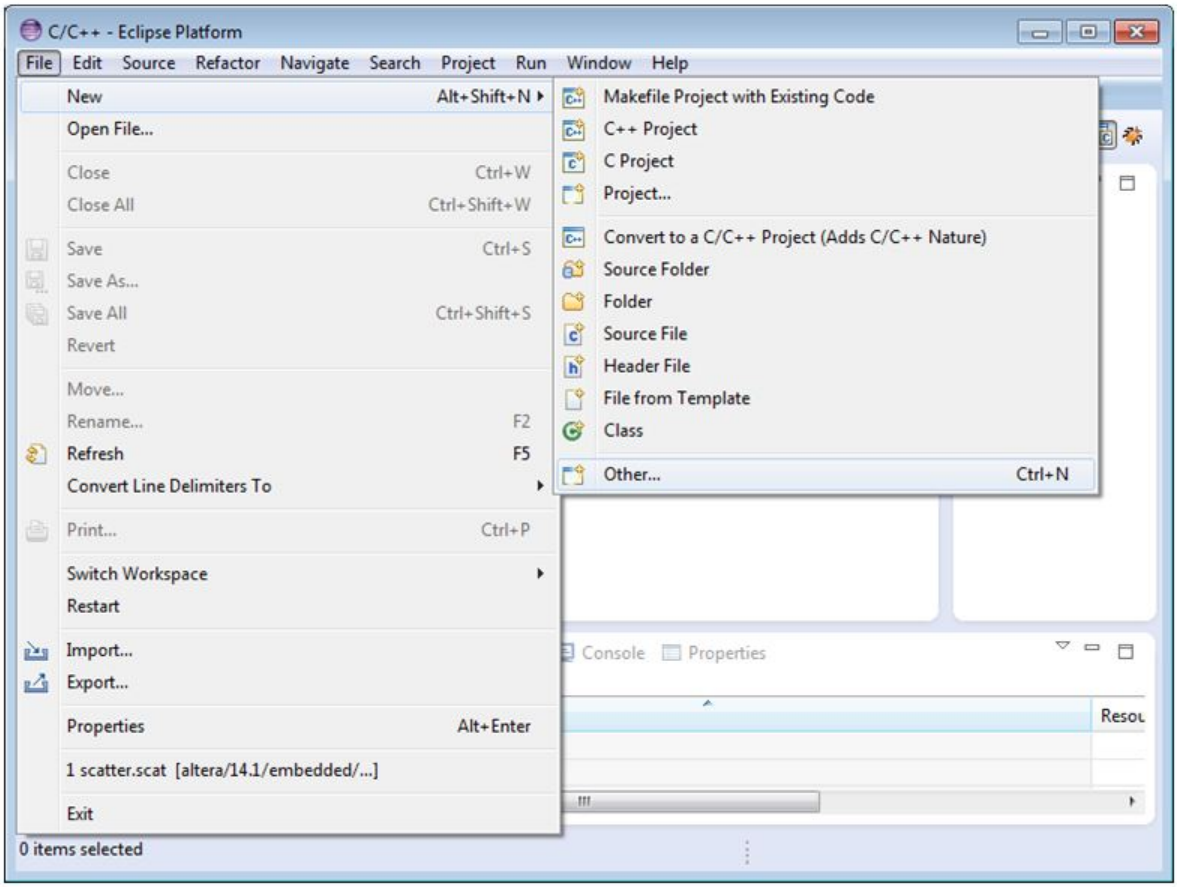



## Linker 脚本(Linker Script)

ARM DS-5 AE 提供一个可视化工具来帮助创建 linker script。

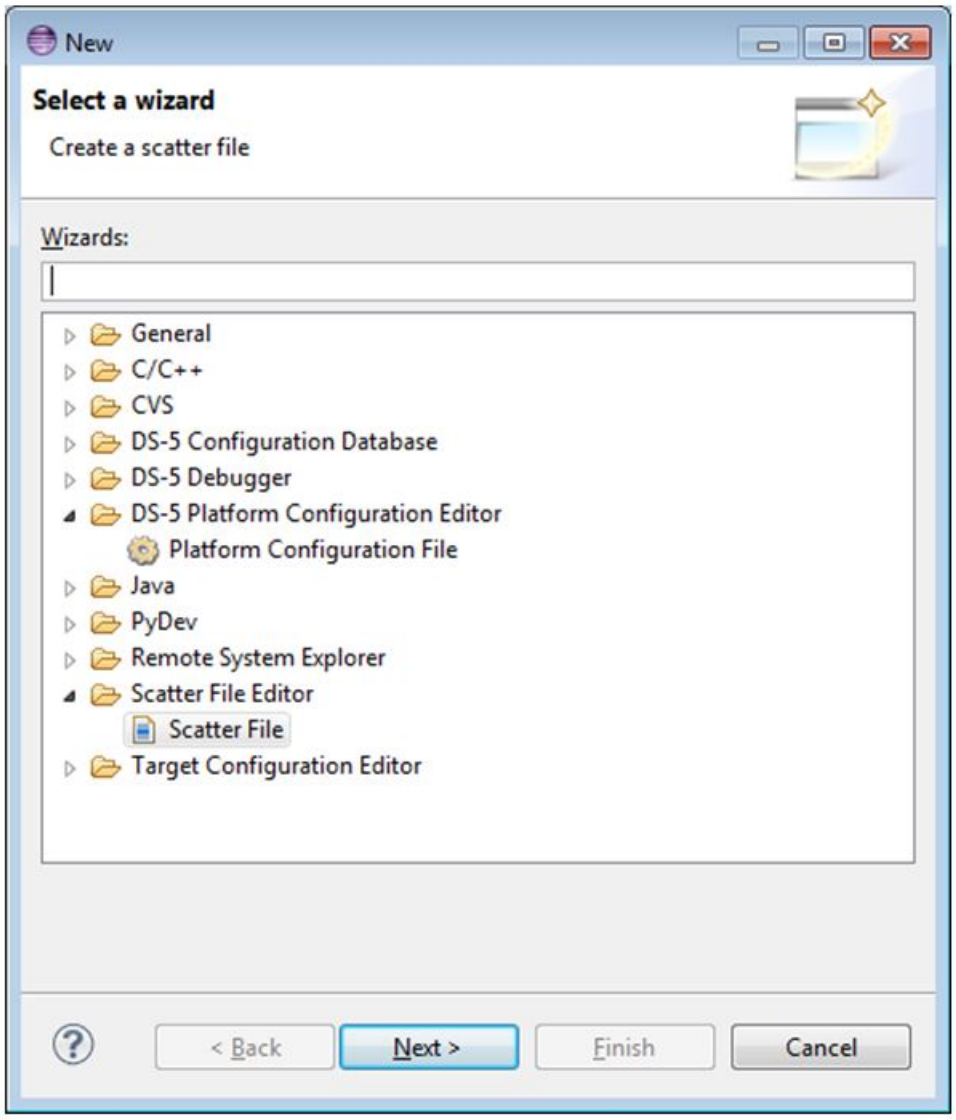
1. 去到 **File** > **New** > **Other...**

**图 6-10: 创建一个 Linker Script**



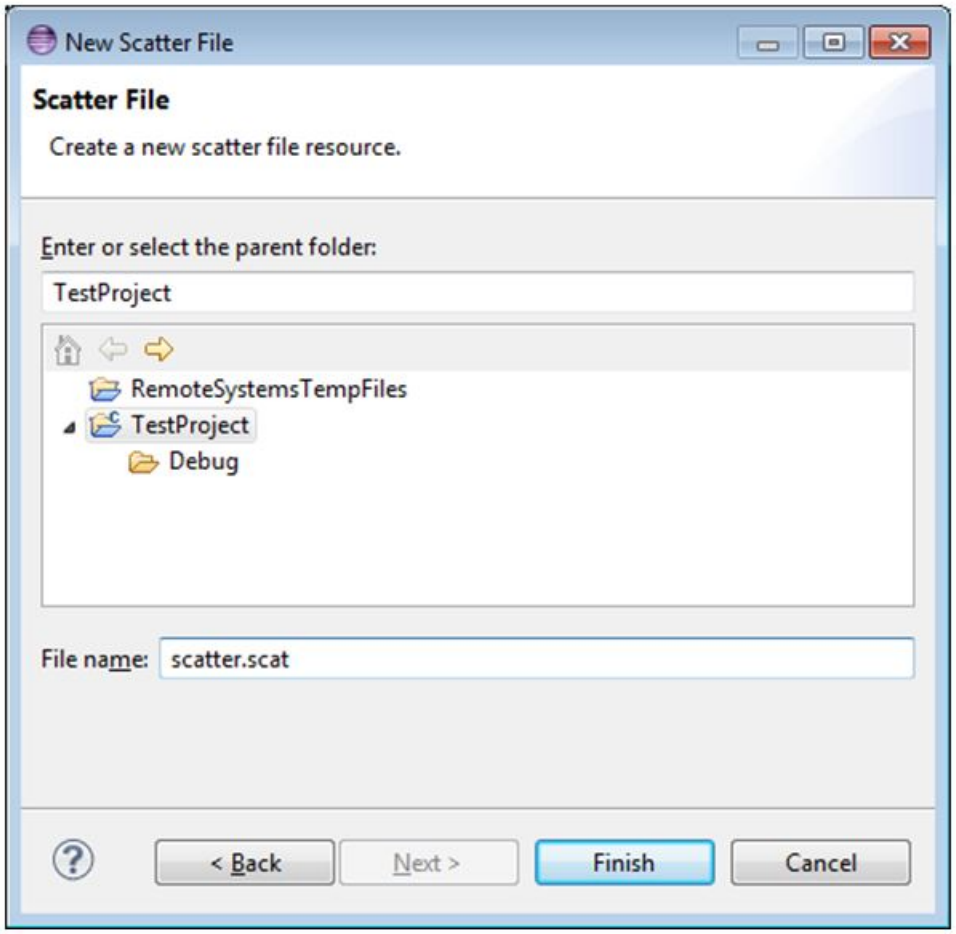

2. 选择 **Scatter File Editor** > **Scatter File**，点击 **Next**。

**图 6-11: 创建一个 Scatter 文件**



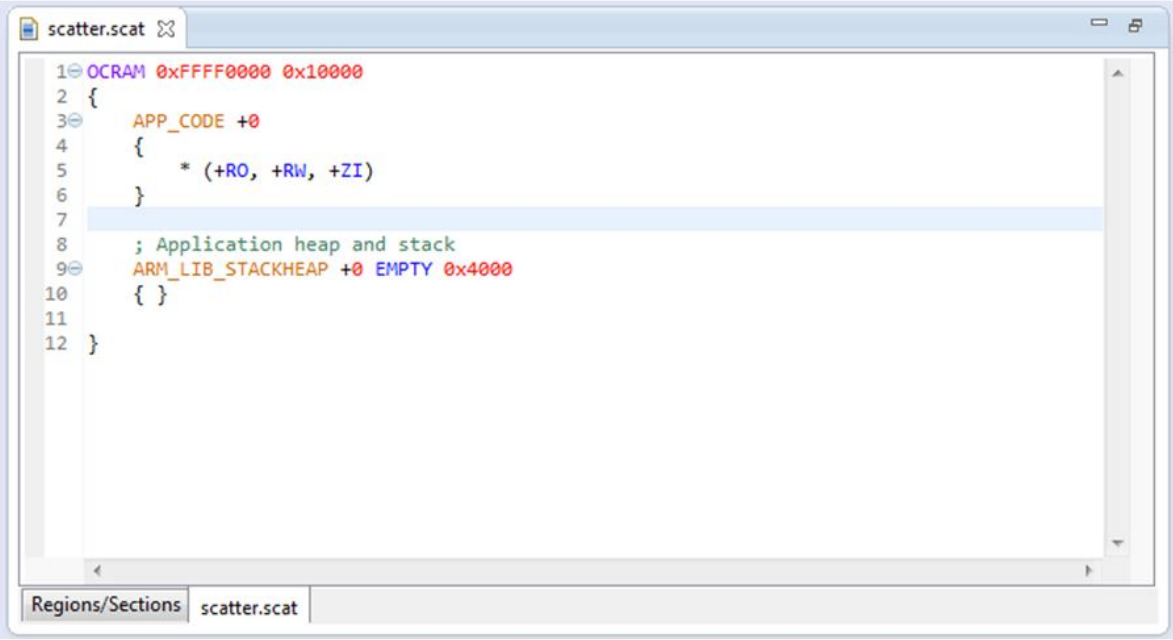

3. 选择新文件的位置，输入文件名，点击 **Finish**。

**图 6-12: 创建一个新的 Scatter 文件资源**



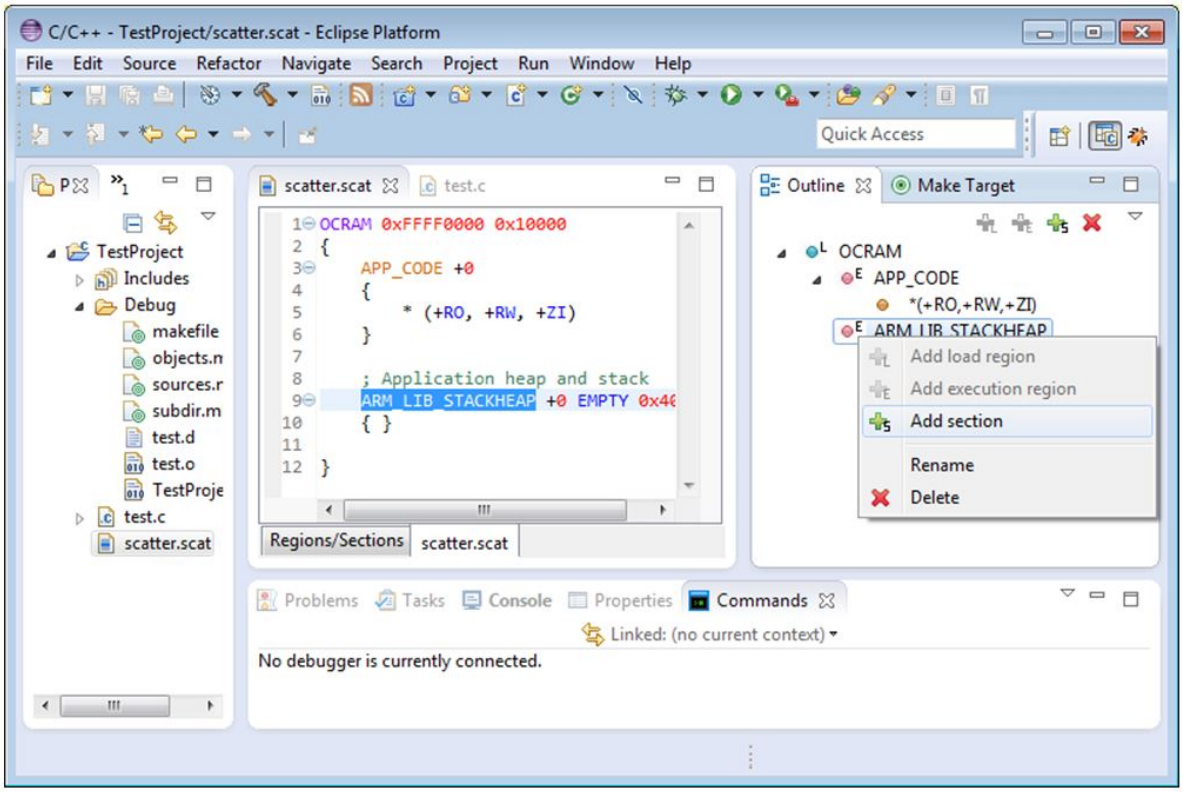

4. 如下例所示，您可以直接编辑 linker script 文件。

**图 6-13: Linker Script 示例**




5. 提供使用 Outline 视窗中的工具编辑此文件。

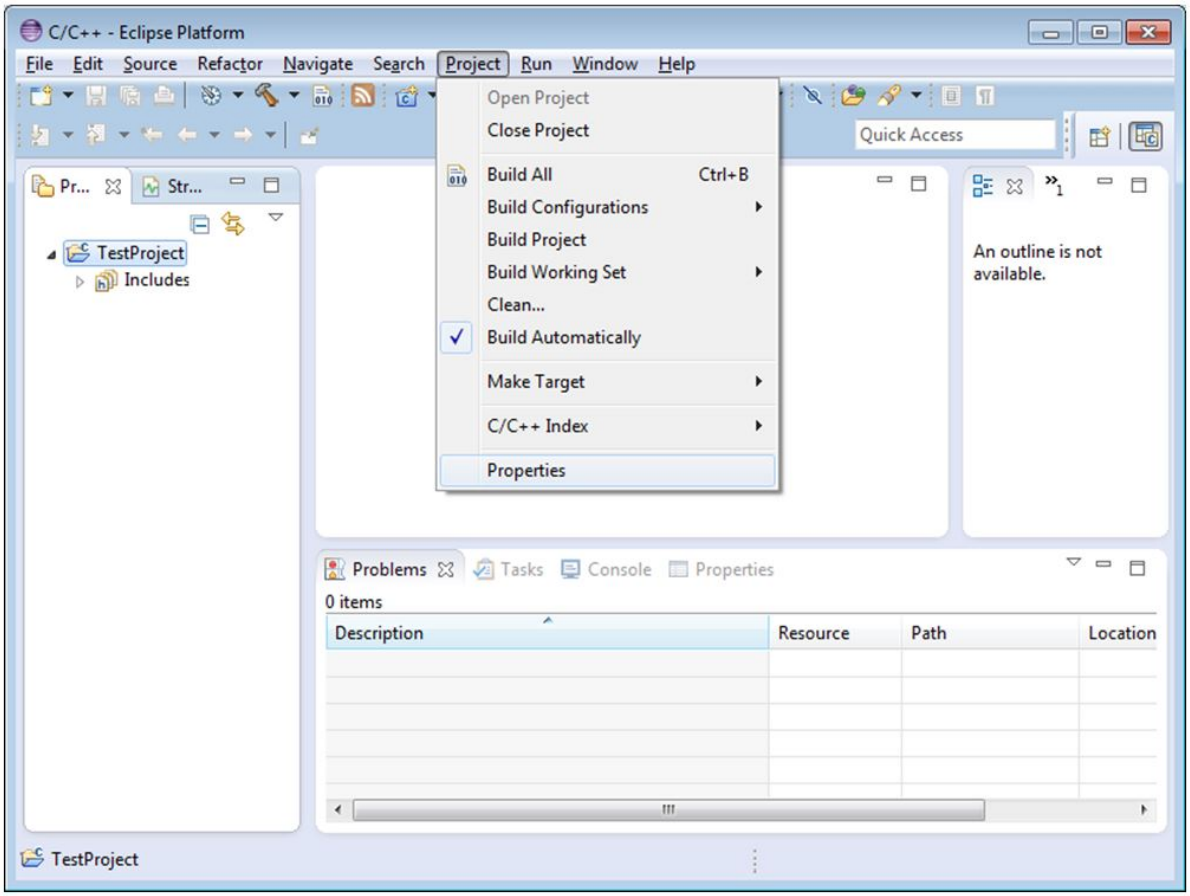
**图 6-14: 使用 Outline 视窗中的工具编辑"Scatter.scat"文件**




## Build 设置

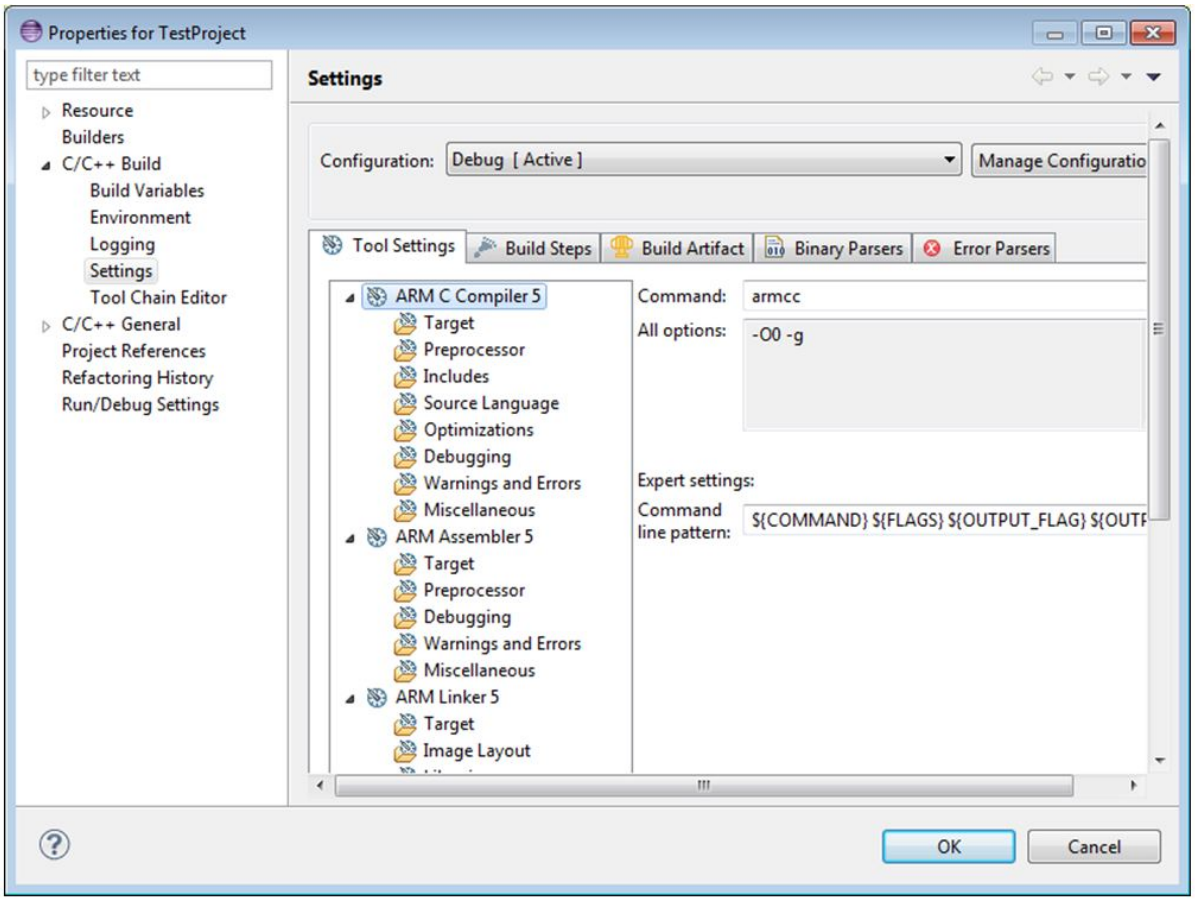
1. 工程一旦创建，就可以通过 **Project** > **Properties** 访问此工程的属性。

**图 6-15: 工程属性**




2. 然后，在 **Project Properites** 窗口中可以通过选择 **C/C++ Build** > **Settings** 来访问"Compilation"设置。

**图 6-16: 工程设置**




build 设置包括所有工具的详细设置：

- Compiler
- Assembler
- Linker

"ARM Compiler 逻辑工程管理入门"包含一个如何在 Altera SoC 开发板上创建一个工程，编译并运行的完整指南的链接。

**相关链接**

入门指南 (第 5-1 页)

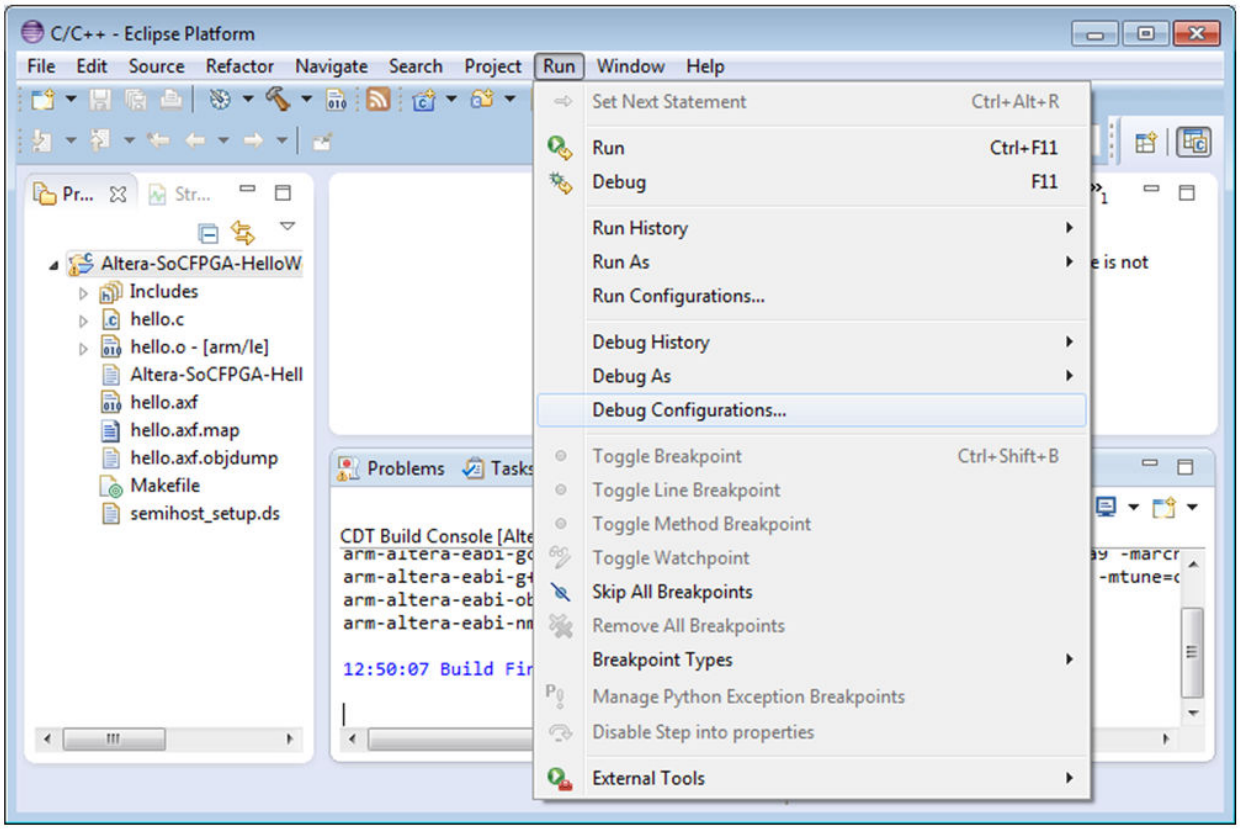
# 调试

ARM DS-5 AE 具有多种调试特性。

## 访问调试配置(Debug Configurations)

用于调试的设置存储在 **Debug Configuration** 中。从 **Run** > **Debug Configurations** 菜单访问 **Debug Configurations** 窗口。

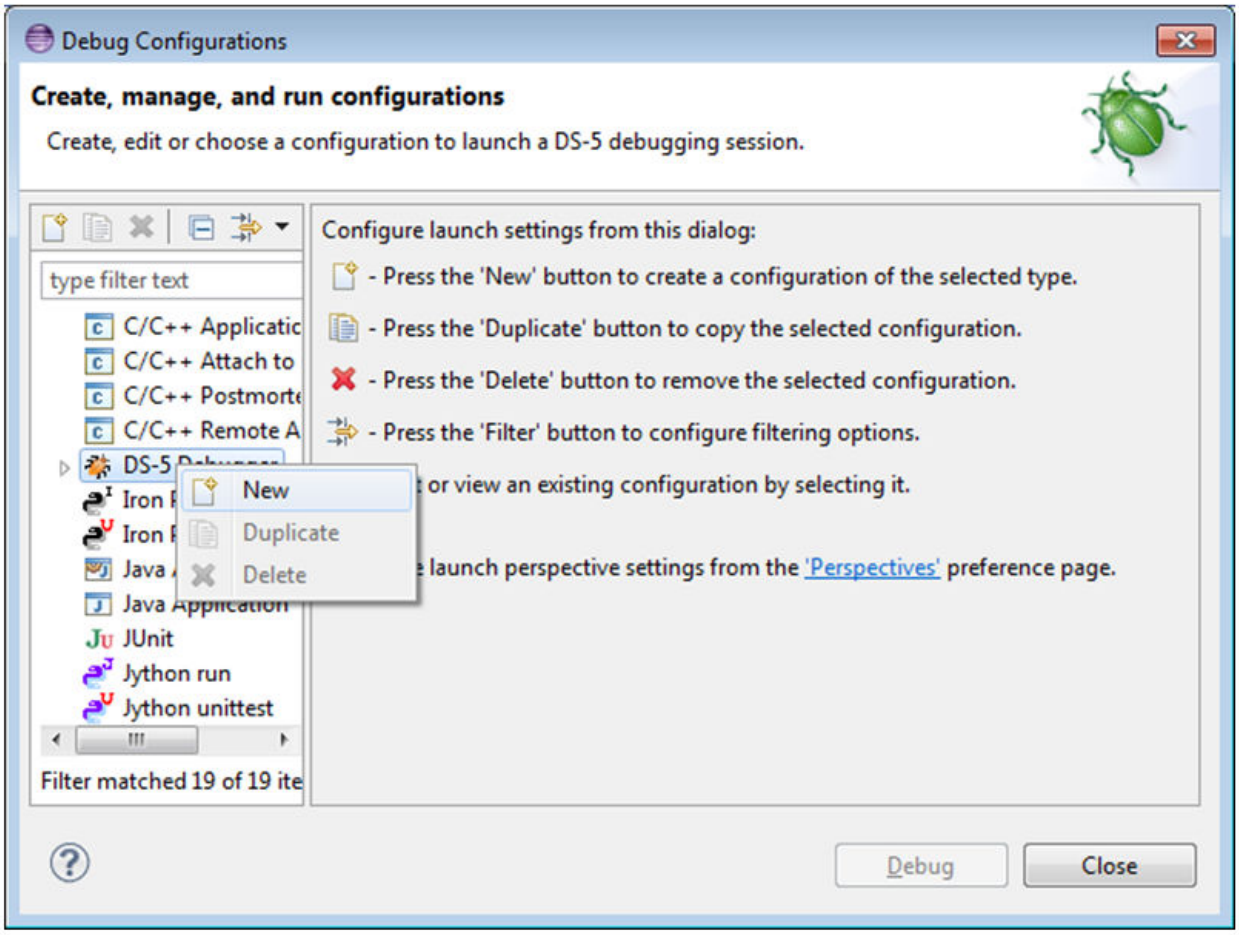
**图 6-17: 访问调试配置(Debug Configuration)**




## 创建一个新的调试配置

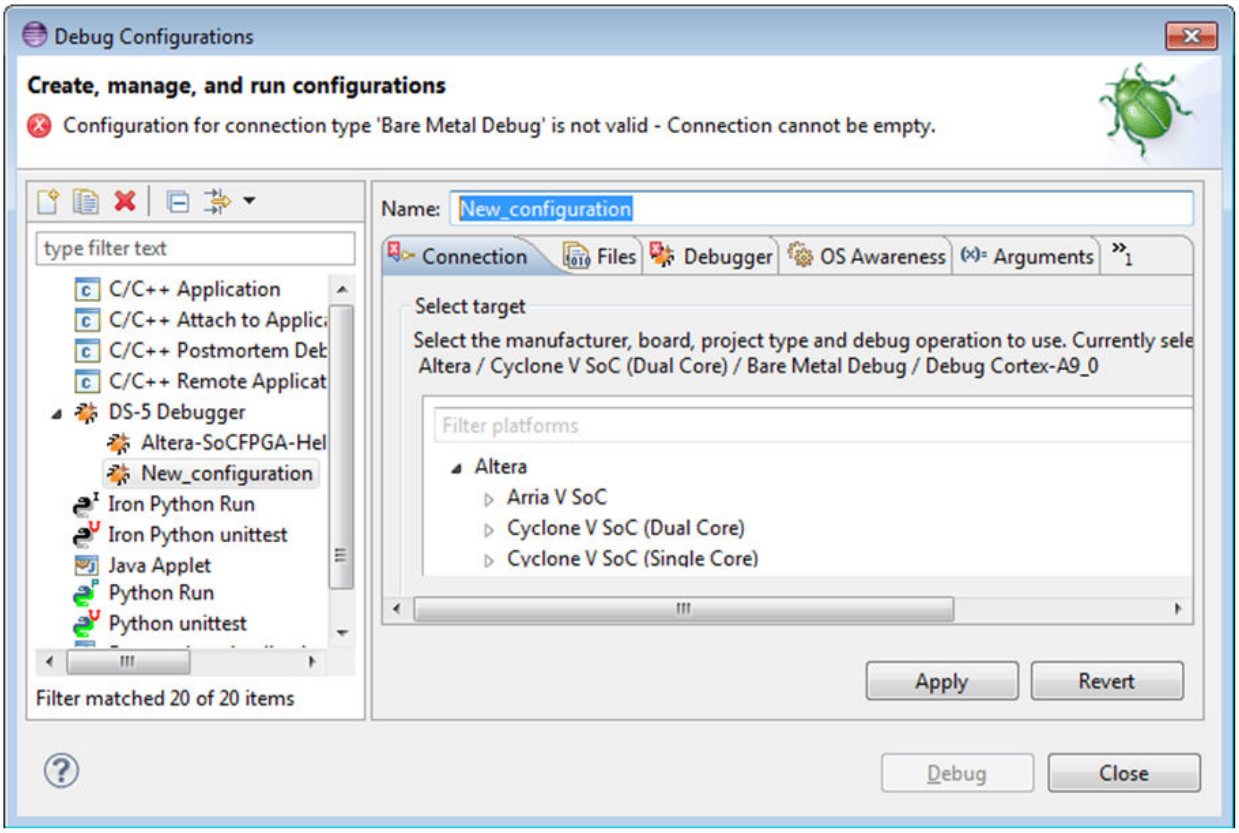
在 **Debug Configurations** 窗口中的左侧面板中选择 **DS-5 Debugger** 作为配置的类型，然后右击选择 **New** 菜单选项来创建一个 **Debug Configuration**。

**图 6-18: 创建新的调试配置**




在 ARM DS-5 AE 中，您可以对配置分配一个可编辑的默认名称。

**图 6-19: 重命名调试配置**




## 调试配置选项

此部分列出了 **Debug Configuration** 选项，使您能够对一个工程指定所需的调试选项：

- Connection Options(连接选项)
- File Options(文件选项)
- Debugger Options(调试器选项)
- RTOS Awareness(RTOS 意识)
- Arguments(自变量)
- Environment(环境)
- Event Viewer(事件查看器)

**相关链接**

- 入门指南
  提供如何使用 ARM DS-5 AE 调试功能的实例。
- 在线 ARM DS-5 文档
  请参考 DS-5 参考文档来获得完整详细的信息。

### Connection Options(连接选项)

**Connection** 标签使用户能够选择所需目标。以下目标用于 Altera 平台：

Arria 10 SoC:

- 裸机调试
  - 调试 Cortex-A9_0
  - 调试 Cortex-A9_1
  - 调试 Cortex-A9x2 SMP
- Linux 应用程序调试
  - 连接到已经正在运行的 **gdbserver**
  - 下载并调试应用程序
  - 开始 **gdbserver** 并调试目标驻留应用程序
- Linux Kernel 和/或器件驱动程序调试
  - 调试 Cortex-A9_0
  - 调试 Cortex-A9_1
  - 调试 Cortex-A9x2 SMP

Arria V SoC:

- 裸机调试
  - 调试 Cortex-A9_0
  - 调试 Cortex-A9_1
  - 调试 Cortex-A9x2_SMP
- Linux 应用程序调试
  - 连接到已经正在运行的 **gdbserver**
  - 下载并调试应用程序
  - 开始 **dbgserver** 并调试目标驻留应用程序
- Linux Kernel 和/或器件驱动程序调试
  - 调试 Cortex-A9_0
  - 调试 Cortex-A9_1
  - 调试 Cortex-A9x2_SMP

Cyclone V SoC (单核):

- 裸机调试
  - 调试 Cortex-A9_0
- Linux 应用程序调试
  - 连接到已经正在运行的 **gdbserver**
  - 下载并调试应用程序
  - 开始 **dbgserver** 并调试目标驻留应用程序
- Linux Kernel 和/或器件驱动程序调试
  - 调试 Cortex-A9_0

Cyclone V SoC (双核):

- 裸机调试
  - 调试 Cortex-A9_0
  - 调试 Cortex-A9_1
  - 调试 Cortex-A9x2_SMP
- Linux 应用程序调试
  - 连接到已经正在运行的 **gdbserver**
  - 下载并调试应用程序
  - 开始 **dbgserver** 并调试目标驻留应用程序
- Linux Kernel 和/或器件驱动程序调试
  - 调试 Cortex-A9_0
  - 调试 Cortex-A9_1
  - 调试 Cortex-A9x2_SMP

Dual Arria V SoC (两个双核 SoC):

- 裸机调试
  - 调试 HPS0 Cortex-A9_0
  - 调试 HPS0 Cortex-A9_1
  - 调试 HPS0 Cortex-A9x2_SMP
  - 调试 HPS1 Cortex-A9_0
  - 调试 HPS1 Cortex-A9_1
  - 调试 HPS1 Cortex-A9x2_SMP
- Linux 应用程序调试
  - 连接到已经正在运行的 **gdbserver**
  - 下载并调试应用程序
  - 开始 **dbgserver** 并调试目标驻留应用程序
- Linux Kernel 和/或器件驱动程序调试
  - 调试 HPS0 Cortex-A9_0
  - 调试 HPS0 Cortex-A9_1
  - 调试 HPS0 Cortex-A9x2_SMP
  - 调试 HPS1 Cortex-A9_0
  - 调试 HPS1 Cortex-A9_1
  - 调试 HPS1 Cortex-A9x2_SMP

Dual Cyclone V SoC (两个双核 SoC):

- 裸机调试
  - 调试 HPS0 Cortex-A9_0
  - 调试 HPS0 Cortex-A9_1
  - 调试 HPS0 Cortex-A9x2_SMP
  - 调试 HPS1 Cortex-A9_0
  - 调试 HPS1 Cortex-A9_1
  - 调试 HPS1 Cortex-A9x2_SMP
- Linux 应用程序调试
  - 连接到已经正在运行的 **gdbserver**
  - 下载并调试应用程序
  - 开始 **dbgserver** 并调试目标驻留应用程序
- Linux Kernel 和/或器件驱动程序调试
  - 调试 HPS0 Cortex-A9_0
  - 调试 HPS0 Cortex-A9_1
  - 调试 HPS0 Cortex-A9x2_SMP
  - 调试 HPS1 Cortex-A9_0
  - 调试 HPS1 Cortex-A9_1
  - 调试 HPS1 Cortex-A9x2_SMP

Android 应用程序调试：

- 本地应用程序/库调试
  - 通过 `gdbserver` 的 APK 本地库调试
    - 连接到一个正在运行的 Android 应用程序
    - 下载并调试一个 Android 应用程序

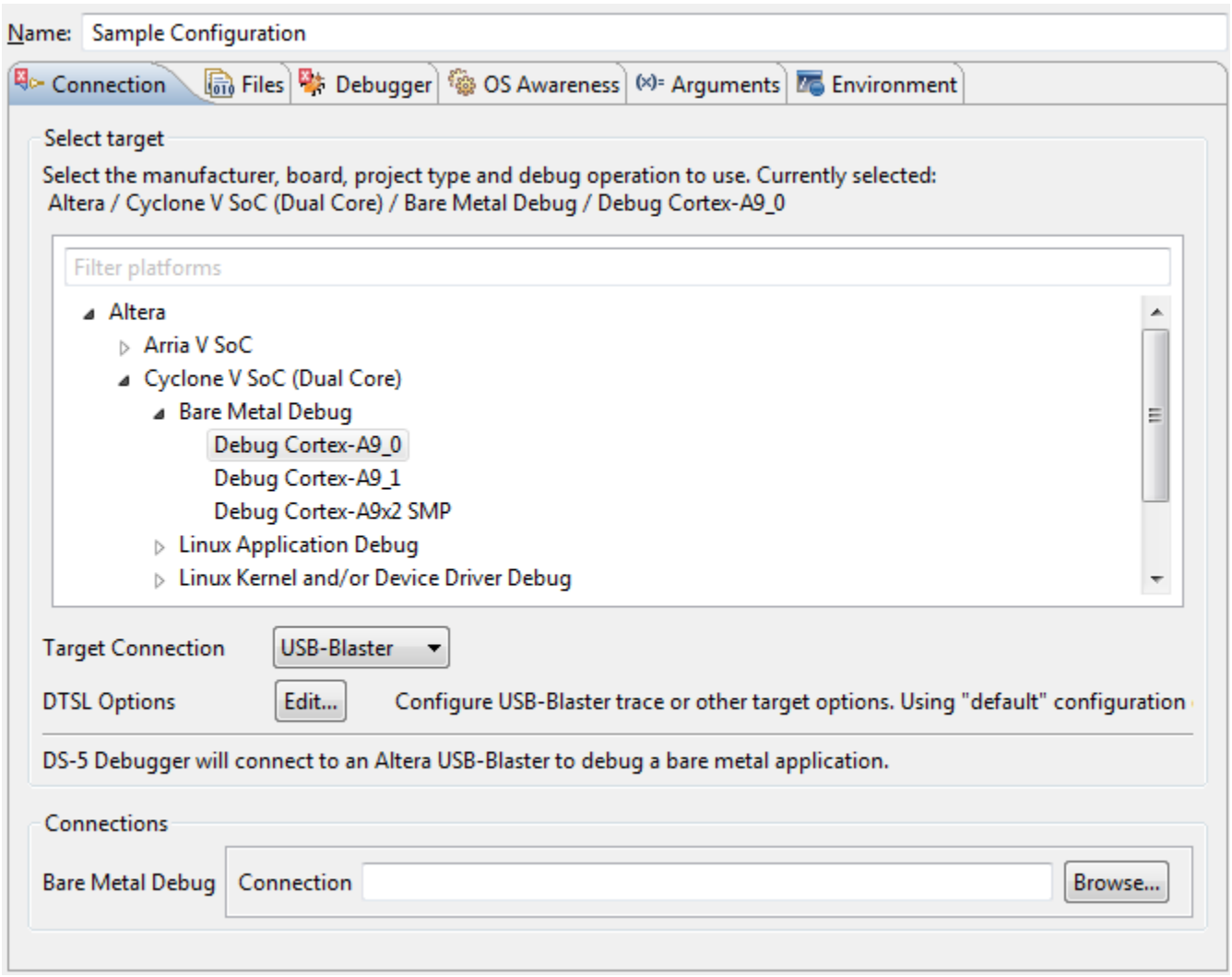
Linux 应用程序调试：

- 应用程序调试
  - 通过 `gdbserver` 的连接
    - 连接到已经正在运行的 **gdbserver**
    - 下载并调试应用程序
    - 开始 **dbgserver** 并调试目标驻留应用程序

所选的 Target 不同，Connections 面板看上去也不同。对于 Bare Metal Debug 和 Linux Kernel and/or Device Driver Debug 目标类型：

- 出现 Target Connection 选项，使用户能够选择连接到目标的类型。Altera USB-Blaster 和 DSTREAM 是两个最常用的选项。
- 出现 DTSL 选项，使用户能够配置 Debug and Traces Services Layer (稍后详细介绍)。
- 出现 Connections Browse 按钮，使用户能够浏览和选择连接的特定实例 — Altera USB-Blaster 或者 DSTREAM 实例。

**图 6-20: Bare-metal 和 Linux Kernel and/or Device Driver Debug 的连接选项**




对于 **Linux Application Debug** 目标，所选的连接类型不同，连接参数也不同。下面两个图显示了选项。

图 6-21: Linux Application Debugging – Connect to a Running GDB Server

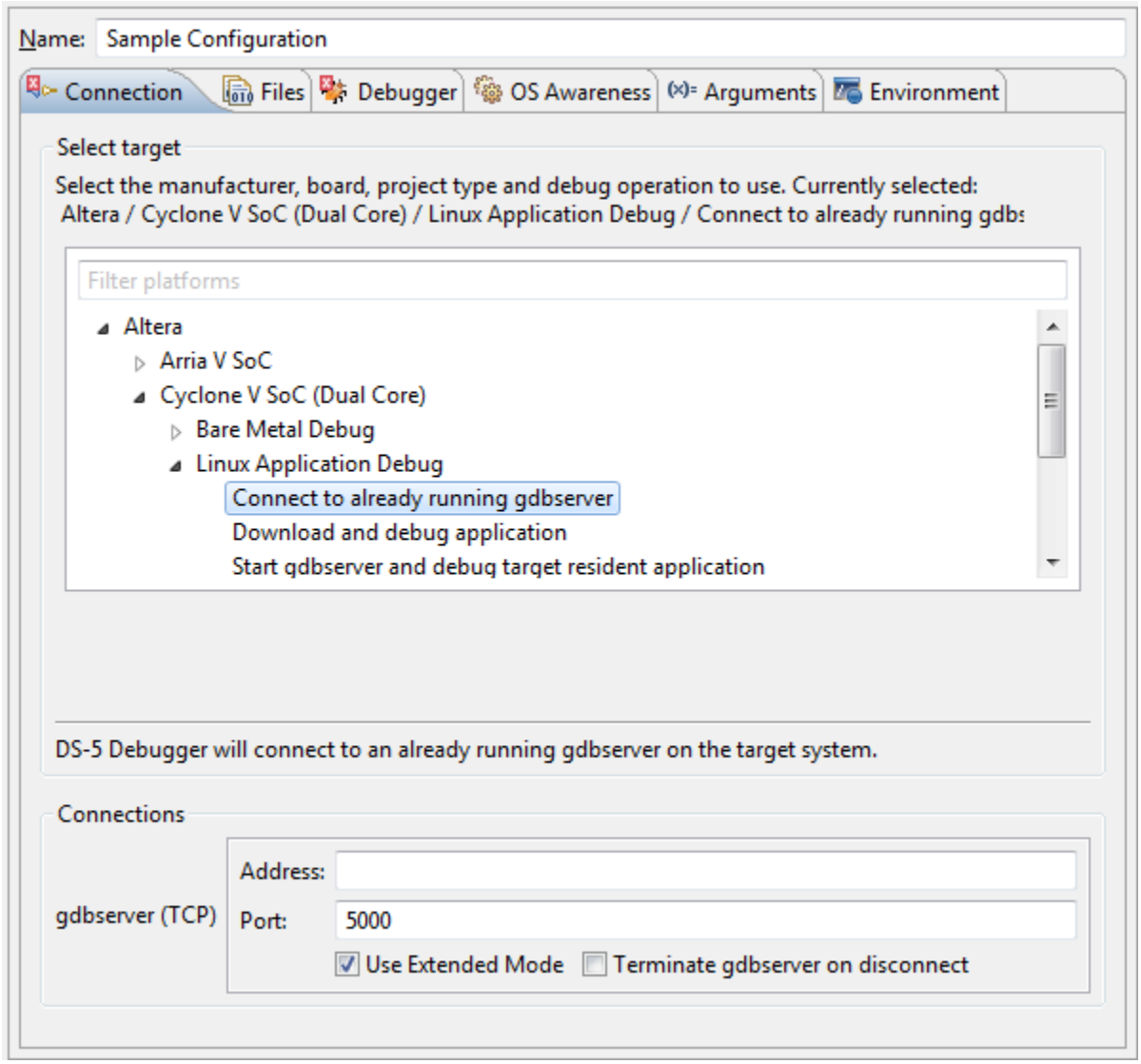
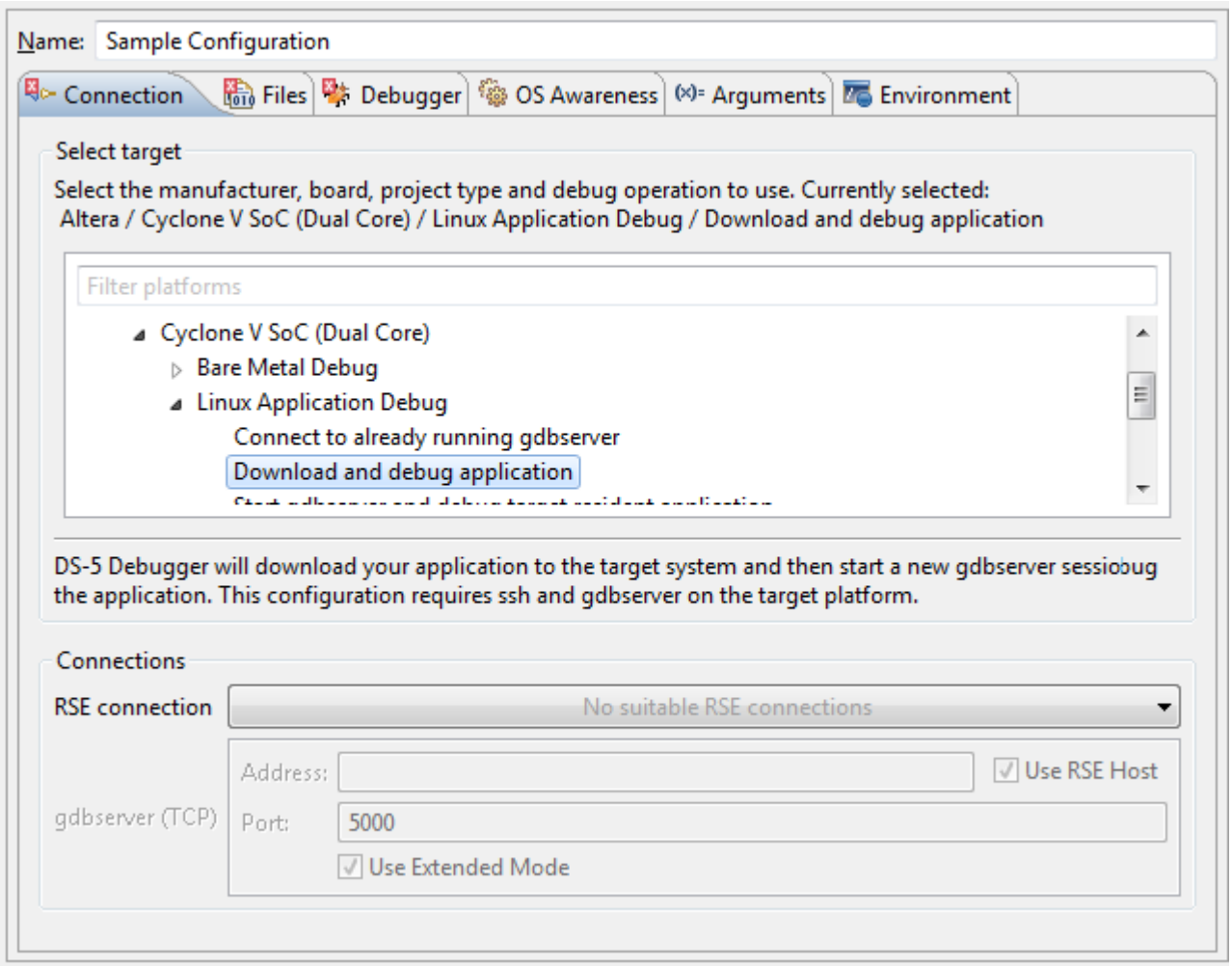


**图 6-22: Linux Application Debugging – Download And Debug Application**




**注意:** 对于 **Linux Application Debug**，需要在 **Remote System Explorer** 中配置 **Connection**，如 *Linux 应用程序调试入门*中所示。

**相关链接**

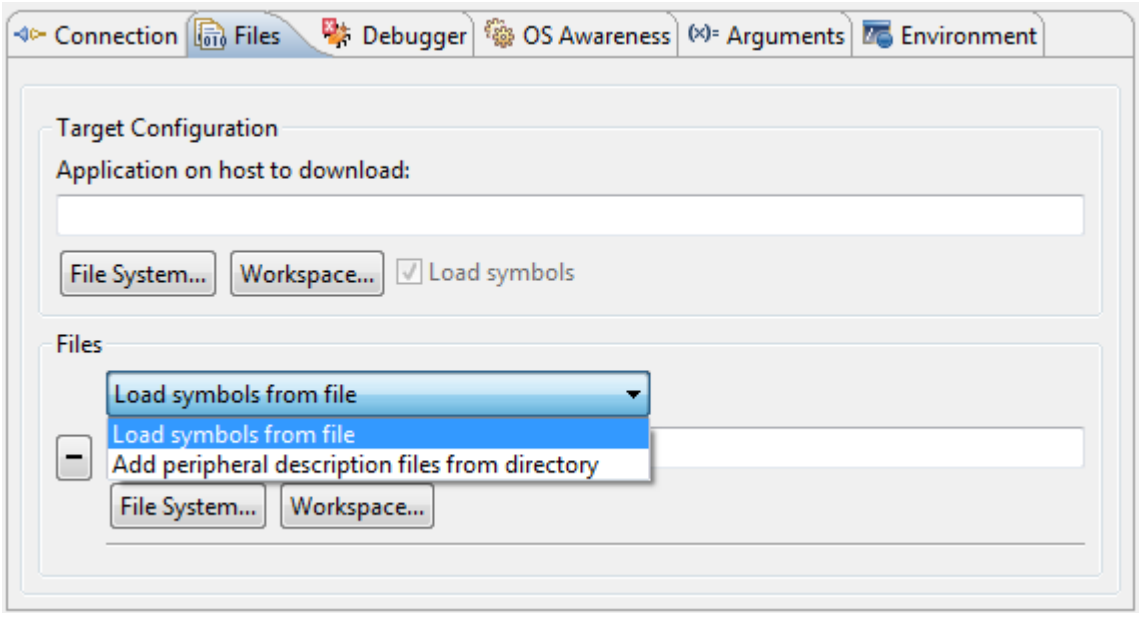
- **DTSL 选项** (第 6-31 页)
  关于 Connections 标签上的选项的详细信息，请参考 DTSL Options 部分。
- **Debugger 选项** (第 6-28 页)
- 入门指南 (第 5-1 页)

## Files 选项

**Files** 标签支持对下面设置进行配置：

- **Application on host to download** – 要下载到目标的应用程序的文件名称。可以直接在编辑框中输入文件名，也可以在 **Workspace** 或 **File System** 中浏览到。
- **Files** – 包含一组文件。使用“+”按钮添加一个文件，使用“–“按钮删除文件。每个文件的类型可以是下面其中一种：
  - **Load symbols from file** – 调试器将使用此文件加载符号。
  - **Add peripheral description files from directory** – 调试器从存储在此目录的.SVD 文件加载外设寄存器描述。SVD 文件产生自硬件工程的编译。

**图 6-23: Files 设置**

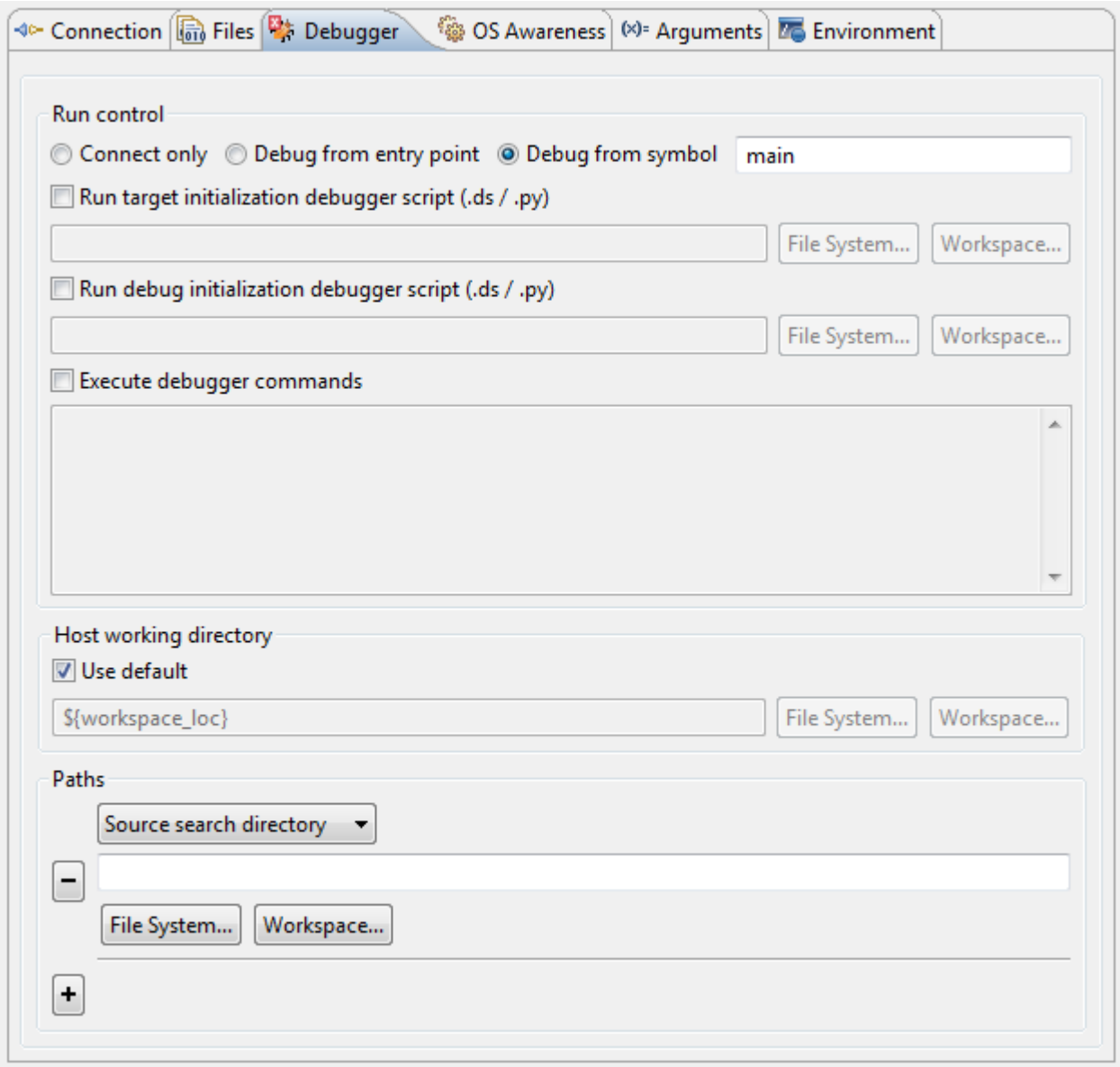



## Debugger 选项

**Debugger** 标签提供以下可配置选项：

- **Run Control** 选项
  - 仅连接，从入口点或用户定义的符号进行调试的选项
  - 运行用户指定的目标初始化脚本的选项
  - 运行用户指定的调试初始化脚本的选项
  - 执行用户定义的调试器命令的选项
- **Host working directory** – 被 semihosting 使用
- **Paths** – 使用户能够输入多个路径，用于调试器搜索资源。使用“+”和“-“按钮添加和删除路径。

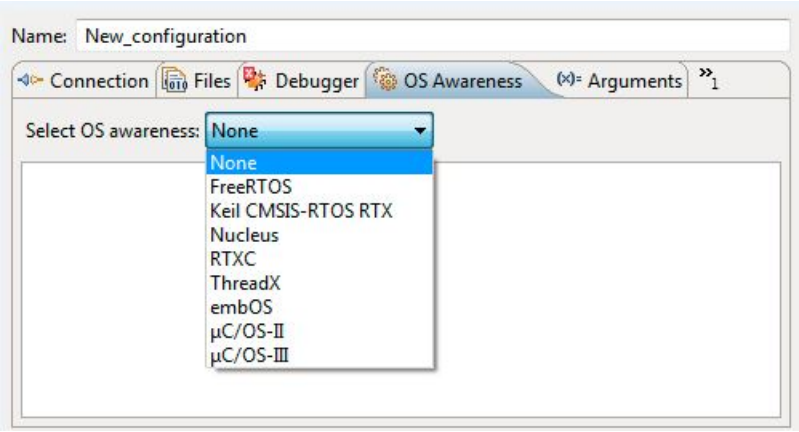
**图 6-24: Debugger 设置**




## RTOS Awareness

**RTOS Awareness** 标签使用户能够对调试器使能 RTOS awareness。

**图 6-25: RTOS Awareness 设置**




**相关链接**

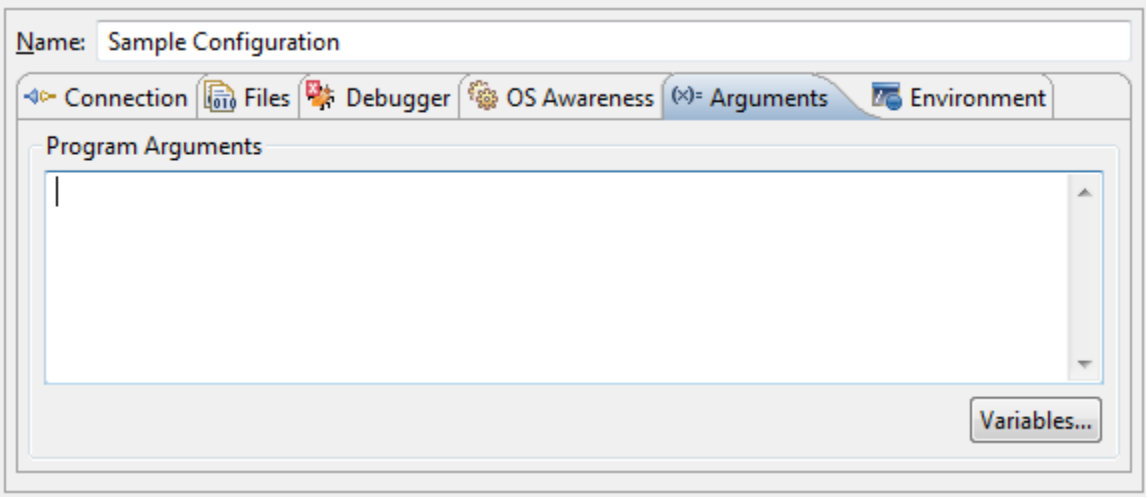
**Keil 网站**

关于 RTOS Awareness 的详细信息，请参考 KeilTM™网站上的 Embedded Development Tools 页面。

## Arguments

**Arguments** 标签使用户能够输入文本格式的程序参数。

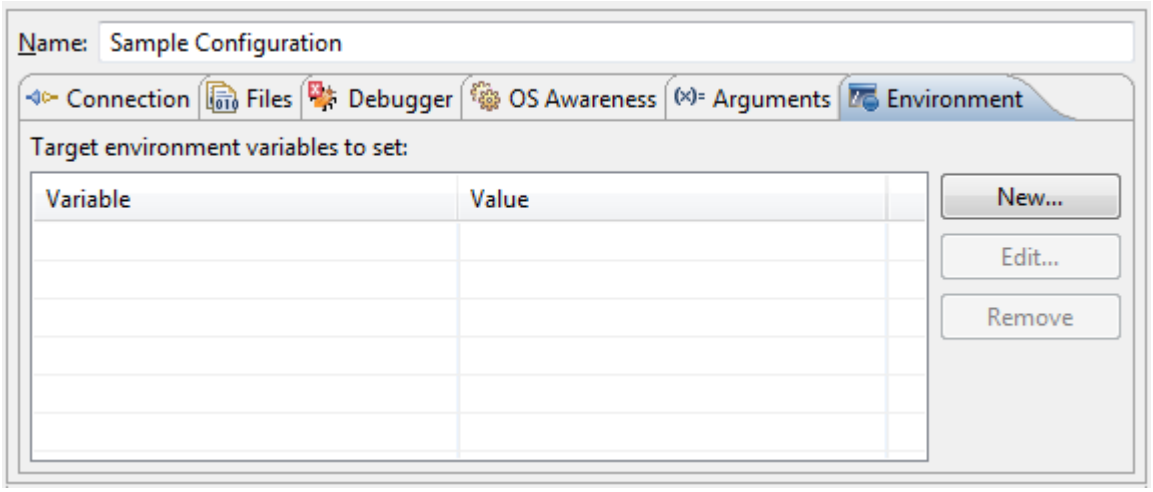
**图 6-26: Arguments 设置**




## Environment

**Environment** 标签使用户能够输入要执行程序的环境变量。

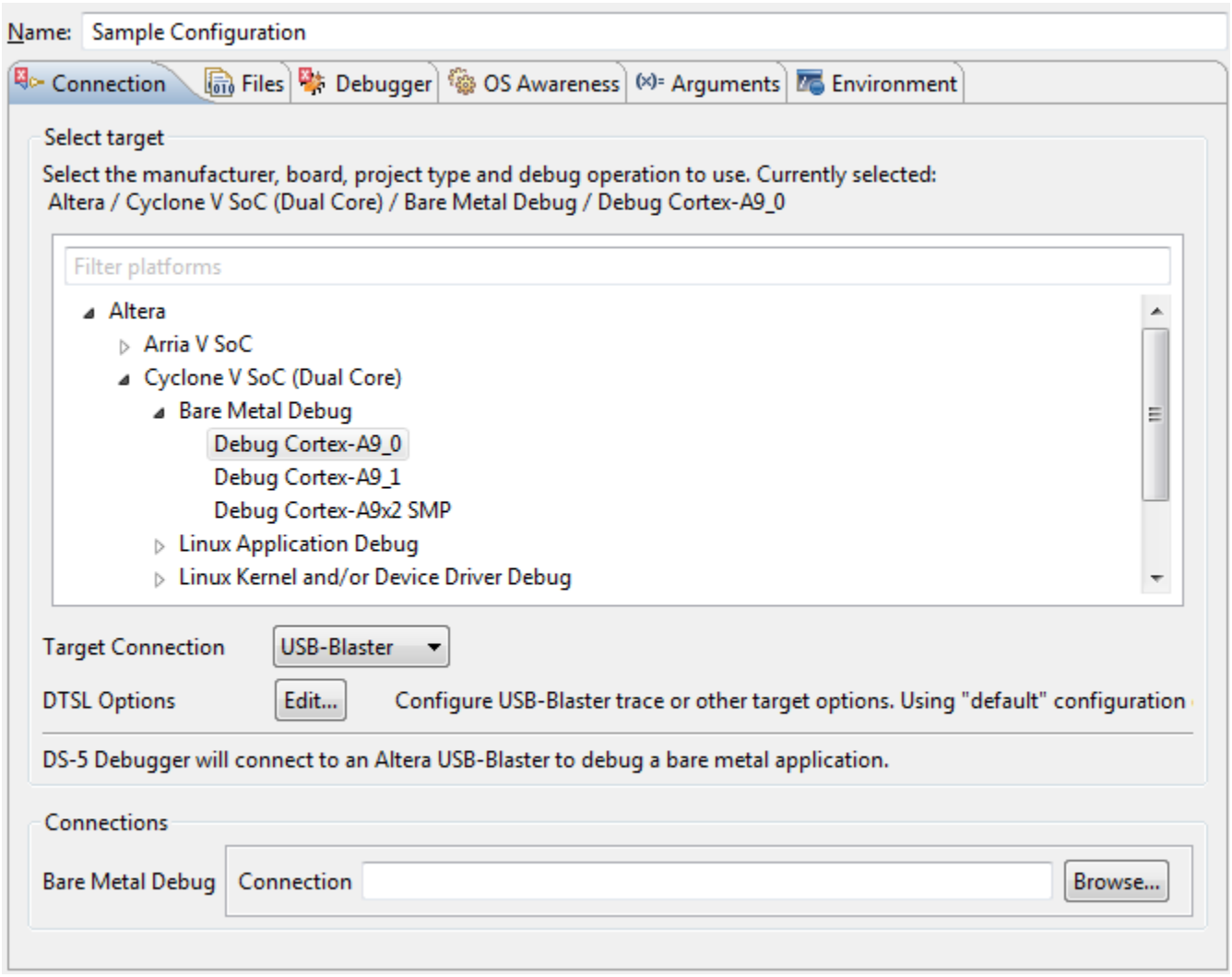
**图 6-27: Environment 设置**




## DTSL 选项

Debug and Trace Services Layer (DTSL)提供跟踪功能。要配置跟踪选项，在工程的 **Debug Configuration** 窗口中，在"Connection"标签中点击 **Edit** 按钮打开 **DTSL Configuration** 窗口。

**图 6-28: Debug Configurations - DTSL Options - Edit**



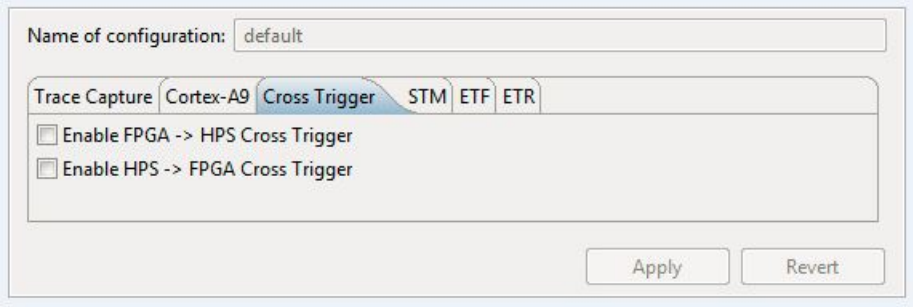

## Cross Trigger 设置

**Cross Trigger** 标签允许 SoC FPGA 的交叉触发选项的配置。

可用选项如下：

- **Enable FPGA > HPS Cross Trigger** – 用于使能从 FPGA 到 HPS 的触发
- **Enable HPS > FPGA Cross Trigger** – 用于使能从 HPS 到 FPGA 的触发

**图 6-29: DTSL Configuration Editor - Cross Trigger**



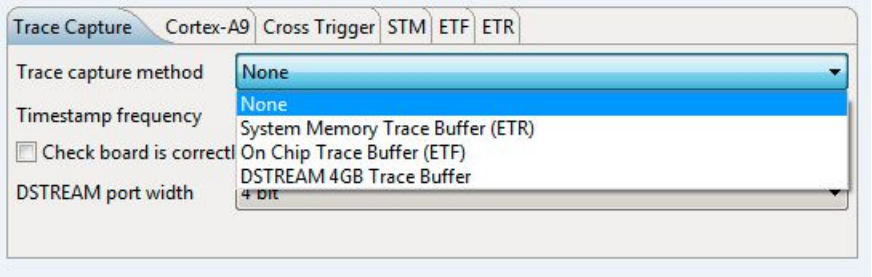

## Trace Capture 设置

**Trace Capture** 标签用于选择跟踪信息的目的地。如简介中所述，目的地可以是下面其中一个：

- **None** – 表示跟踪被禁止
- **ETR** – 使用 HPS 可访问的任何存储缓冲器
- **ETF** – 使用 32KB 片上跟踪缓冲器
- **DSTREAM** – 使用位于 DSTREAM 中的 4GB 缓冲器

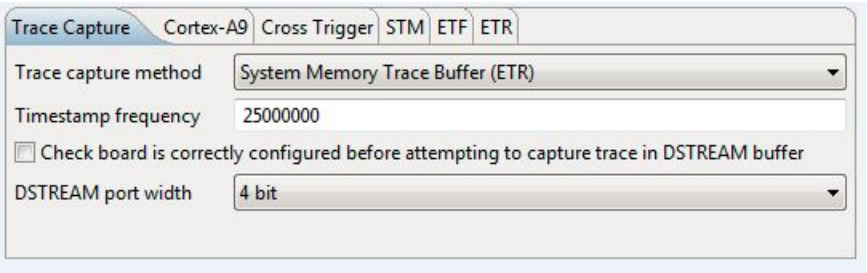
仅当 **Debug Configuration** 中的 **Target** 连接选作 **DSTREAM** 时 DSTREAM 选项才可使用。

**图 6-30: DTSL Configuration Editor - Trace Capture > Trace Capture Method**




**Trace Buffer** 标签提供选择时间戳频率(timestamp frequency)的选项。

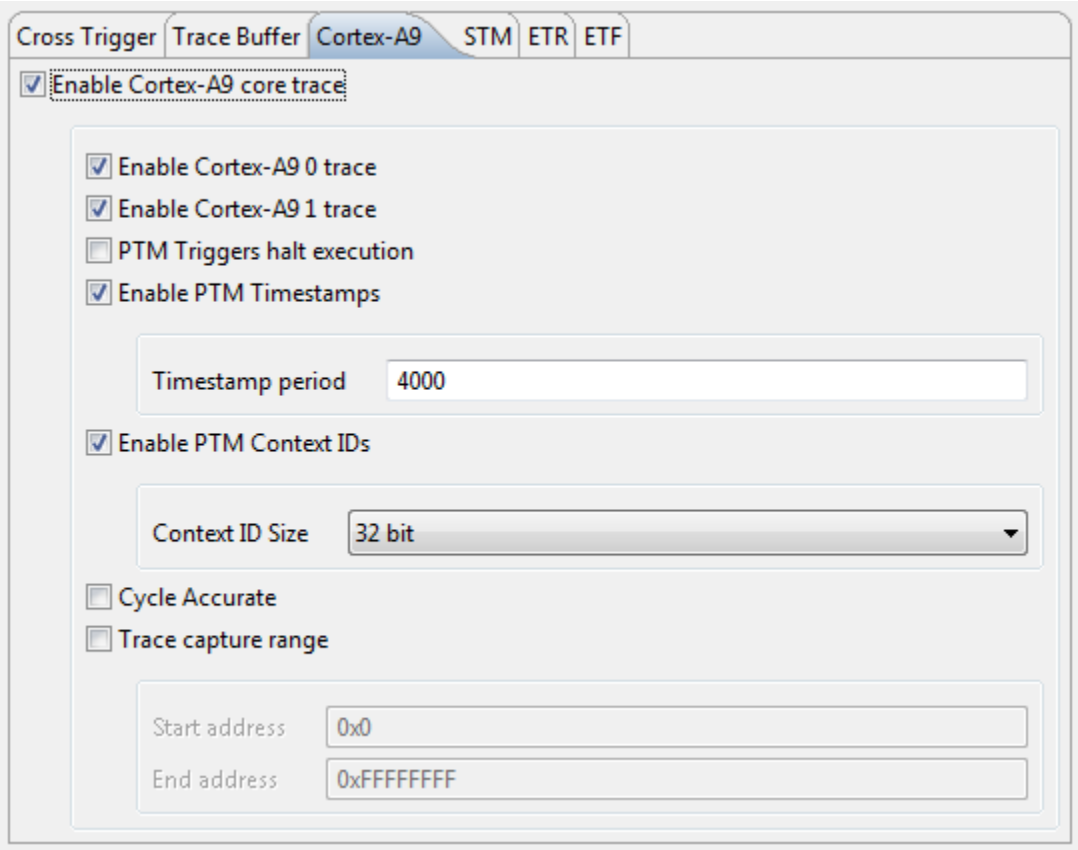
**图 6-31: DTSL Configuration Editor - Trace Capture > Timestamp Frequency**




## Cortex-A9 设置

**Cortex-A9** 标签用于选择内核跟踪选项。

**图 6-32: DTSL Configuration Editor - Cortex-A9**



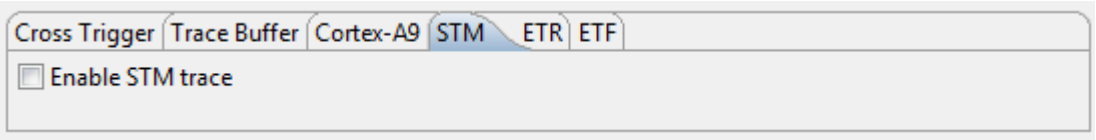

可使用下面的 **Core Tracing Options**:

- **Enable Cortex-A9 0 core trace** – 使能对 core #0 的跟踪
- **Enable Cortex-A9 1 core trace** – 使能对 core #1 的跟踪
- **PTM Triggers halt execution** – 跟踪时导致执行中止
- **Enable PTM Timestamps** – 使能时间戳
- **Enable PMT Context IDs** – 使能跟踪 context ID
- **Context ID Size** – 选择 8-、16-或 32-bit context ID。仅当 Context ID 使能时使用。
- **Cycle Accurate** – 创建周期精确跟踪
- **Trace capture range** – 使能仅跟踪某一地址间隔
- **Start Address, End Address** – 定义跟踪地址间隔(仅当 Trace Capture Range 使能时使用)

## STM 设置

**STM** 标签使您能够配置 System Trace Macrocell (STM)

**图 6-33: DTSL Configuration Editor - STM**




只有一个选项可用：

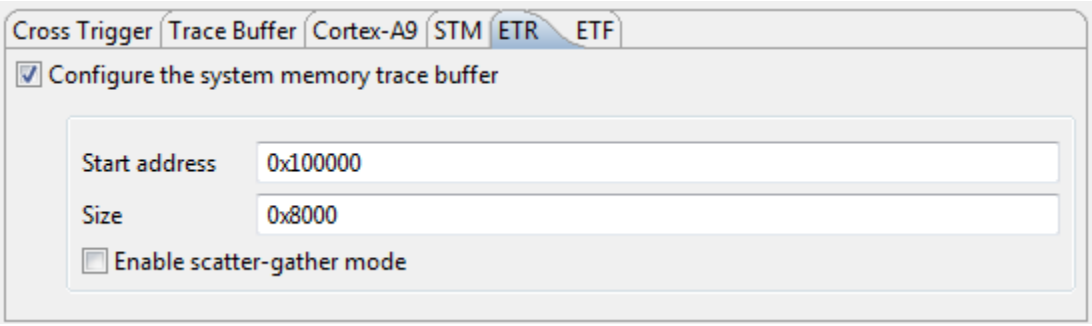
- **Enable STM Trace** – 使能 STM 跟踪。

## ETR 设置

**ETR** 设置用于 Embedded Trace Router (ETR)设置的配置。

Embedded Trace Router 用于将跟踪信息指引到 HPS 可访问的存储缓冲器中。

**图 6-34: DTSL Configuration Editor - ETR**




可用以下选项：

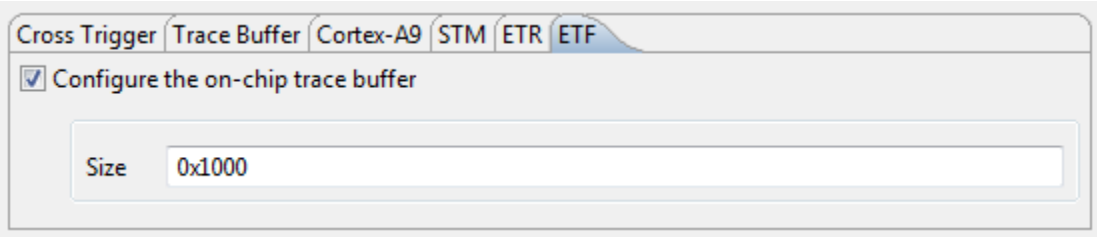
- **Configure the system memory trace buffer** – 如果在 **Trace Capture** 标签上选择 **ETR** 作为跟踪目的地，那么要勾选此选项。
- **Start Address, Size** – 定义系统存储器中跟踪缓冲器位置及容量。
- **Enable scatter-gather mode** – 当 OS 不能保证一块连续的物理内存时使用。scatter-gather 表由操作系统使用器件驱动程序进行设置，EFR 自动读取 scatter-gather 表。

## ETF 设置

**ETF** 标签用于 Embedded Trace FIFO (ETF)设置的配置。

Embedded Trace FIFO 是一个位于 HPS 中的 32 KB 缓冲器，用于存储被调试器检索到的跟踪数据，当跟踪数据通过 ETR 存储在存储器中或者使用 TPIU 将跟踪数据存储在外部 DSTREAM 器件上时也用作弹性缓冲器。

**图 6-35: DTSL Configuration Editor - ETF**




可用以下选项：

- **Configure the on-chip trace buffer** – 如果在 **Trace Capture** 标签上选择 ETF 作为跟踪目的地，那么要勾选此选项。
- **Size** – 定义 ETF 大小。默认大小设置为 0x8000 (32KB)。

# ARM DS-5 AE 文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 更新了 ETF 设置中的默认大小。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# 引导工具用户指南

7

2016.02.17

ug-1137 订阅 反馈

## 引言

所有 Altera SoC 器件的引导流程都包括第二阶段引导加载器(SSBL)。在 Cyclone V 和 Arria V 器件文档中，SSBL 通常称为“预加载器(preloader)”，在 Arria 10 器件文档中称为“引导加载器(bootloader)”。

SSBL 由引导 ROM 加载到片上 RAM (OCRAM)，并负责启动 SDRAM，以及加载和执行引导进程的下一个阶段。

**图 7-1: Cyclone V 和 Arria V 典型的引导流程**

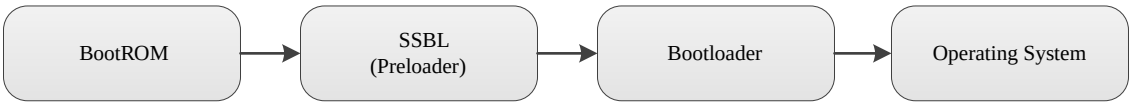



**图 7-2: Arria 10 典型的引导流程**


对于 Cyclone V 和 Arria V 器件，OCRAM 容量为 64 KB，这限制了 SSBL 大小，通常需要一个额外的引导加载器阶段，用于加载操作系统。

对于 Arria 10 器件，不需要额外的引导加载器阶段，因为 OCRAM 容量是 256 KB，并且所需功能包括在 SSBL 中。

本章介绍了用于使能 SSBL 管理的工具：

- **SSBL Support Package Generator** – 使用户能够创建和管理 SSBL
- **SSBL Image Tool ( mkpimage )** – 使用户能够在 SSBL 顶层添加所需引导 ROM 头(header)
- **U-Boot Image Tool ( mkimage )** – 使用户能够在由 SSBL 加载的文件顶层添加所需 SSBL 头(header)

# 第二阶段引导加载器支持封装生成器(Second Stage Bootloader Support Package Generator)

Second Stage Bootloader (SSBL) Support Package Generator 提供一种简单、安全、可靠的方法对 Altera SoC 器件定制 SSBL。

在本文档接下来的部分，SSBL Support Package 表示为“BSP”，SSBL Support Package Generator 表示为“BSP Generator”。

BSP Generator 使您能够执行下面的任务：

- 创建一个新的 BSP
- 报告 BSP 设置
- 修改 BSP 设置
- 生成 BSP 文件

生成的 BSP 包括一个 makefile，可用于构建相应可引导的 Preloader 或 Bootloader 映像。

您可以从 Graphical User Interface 访问 BSP Generator 功能，也可以使用一组命令行工具访问 BSP Generator 功能，命令行工具支持流程的完整脚本。

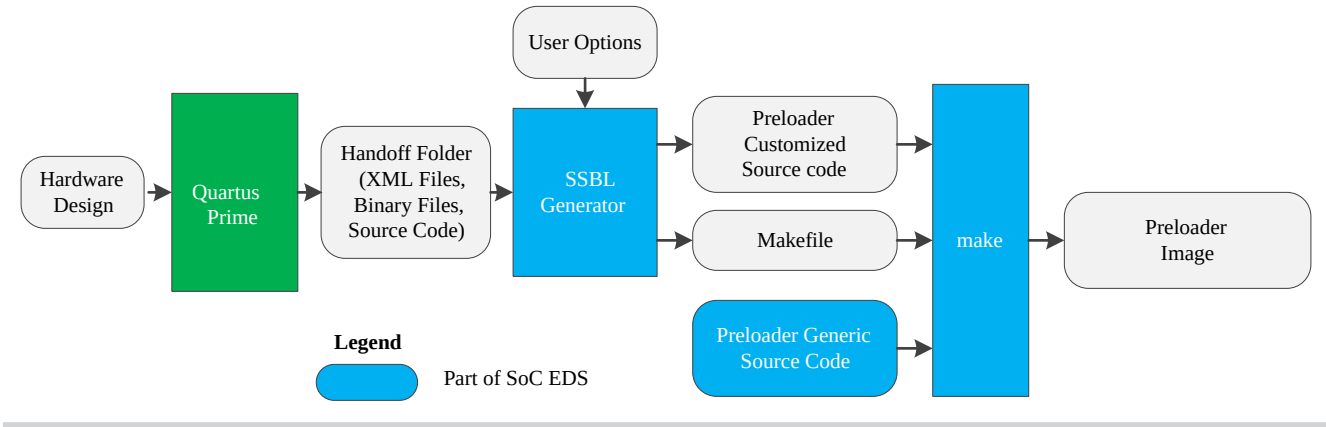
## BSP 生成流程

此部分介绍了 Cyclone V 和 Arria V Preloader 以及 Arria 10 Bootloader 的 BSP 生成流程。这些流程虽然类似，但也有一些重要差异。

### Cyclone V 和 Arria V 流程

对于 Cyclone V 和 Arria V，BSP Generator 通过预加载器通用源文件和特定板级的 SoC FPGA 文件创建一个定制的 BSP。生成器合并硬件设置和用户输入来创建 BSP。BSP 文件包括一个 makefile，以生成预加载器映像。预加载器映像然后可下载到 Flash 器件或 FPGA RAM，用于引导 HPS。

**图 7-3: Arria V/ Cyclone V BSP Generator 流程**




硬件切换信息包含用户在 Qsys 和 Quartus Prime 中创建硬件设计时输入的各种设置，包括：

- HPS 专用管脚的管脚多路复用(pin-muxing)
- HPS 专用管脚的 I/O 设置：
  - 电压
  - 摆率
  - 上拉/下拉
- HPS 外设的状态：
  - 使能
  - 禁用
- HPS 与 FPGA 之间的桥接配置
- 时钟树设置：
  - PLL 设置
  - 时钟分频器设置
  - 时钟 gatting 设置
- DDR 设置：
  - 技术
  - 宽度
  - 速度

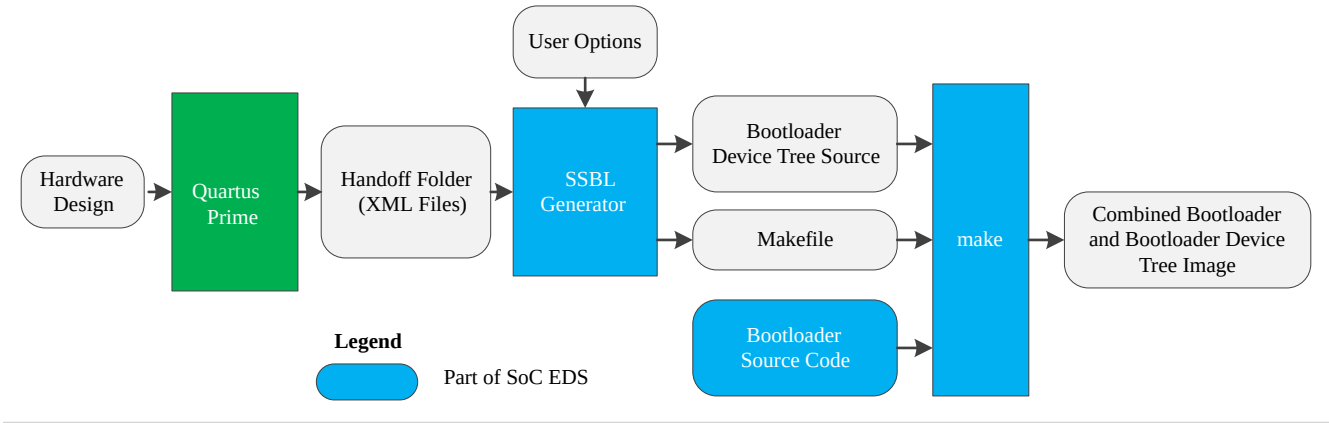
切换设置是 Quartus Prime 编译的输出，位于**<quartus project directory>/hps_isw_handoff/<hps entity name>**目录中(其中**<hps entity name** >是 Qsys 中的 HPS 组件名称)。

您必须更新硬件切换文件，并且每次硬件变更(例如管脚多路复用或管脚约束变更后)对 HPS 有影响时都要重新生成 BSP。

## Arria 10 流程

对于 Arria 10，BSP Generator 创建一个定制的 BSP，其中包括一个 makefile 和 Bootloader Device Tree。由于所有的定制都封装在 Bootloader Device Tree 和 makefile 设置中，因此没有源代码生成。makefile 能用于创建相结合的引导加载器映像，包括引导加载器可执行文件以及引导加载器器件树。相结合的映像然后可下载到 Flash 器件或 FPGA RAM，用于引导 HPS。

**图 7-4: Arria 10 SSBL Support Package Generator 流程**




硬件切换信息包含用户在 Qsys 和 Quartus Prime 中创建硬件设计时输入的各种设置，包括：

- HPS 专用管脚的管脚多路复用(pin-muxing)
- HPS 专用管脚的 I/O 设置：
  - 电压
  - 摆率
  - 上拉/下拉
- 共享管脚的多路复用(pin-muxing)
- HPS 外设的状态：
  - 使能
  - 禁用
- HPS 与 FPGA 之间的桥接配置
- 时钟树设置：
  - PLL 设置
  - 时钟分频器设置
  - 时钟 gatting 设置

切换设置是 Quartus Prime 编译的输出，位于**<quartus project directory>/hps_isw_handoff** 目录中。

每次硬件变更导致以上参数变化时，用户必须运行 BSP Generator，重新生成 Bootloader 器件树。

然而，当硬件设置改变时，用户不必重新编译 Bootloader。仅当改变引导源时才需要重新编译 Bootloader。

## BSP Generator 图像用户界面

您必须按照下面步骤来使用 BSP Generator GUI, **bsp -editor**：

1. 打开嵌入式命令壳。

2. 在嵌入式命令壳中运行 **bsp -editor** 命令来运行 BSP Generator GUI。

3. 要打开和修改现有的 BSP 工程，需要点击 **File > Open** 并找到现有的.bsp 文件。

4. 要创建一个新的 BSP 工程，点击 **File > New HPS BSP** 打开 **New BSP** 对话框。**New BSP** 对话框包含以下设置和参数：

- **Preloader settings directory** – 到硬件切换文件的路径。generator 检查切换文件来验证路径的有效性。
- **Operating system** – 从下面两个选项选择 SSBL 的类型：
  - U-Boot SPL Preloader (Cyclone V/Arria V HPS)
  - U-Boot Bootloader (Arria 10 HPS)
- **Version** – 要使用的 SSBL 版本。此发布仅支持默认的 1.0 版本。
- **Use default locations** – 默认情况下是勾选的，从硬件切换文件夹获得 BSP 的位置。如果需要一个定制路径，则不要勾选此选项。
- **BSP target directory** – 由 generator 创建的新 BSP 文件的目标文件夹。在本文档中表示为**< bsp directory>**。对于 Arria V/ Cyclone V Preloader，默认的目录名是“spl_bsp”，对于 Arria 10 Bootloader，默认的目录名是“uboot_bsp”。当 **Use default locations** 未勾选时可以修改目录名。
- **BSP settings file name** – **. bsp** 文件的位置和文件名，包括 BSP 设置。
- **Additional .tcl scripting** – .tcl 脚本的位置和文件名，用于覆盖默认的 BSP 设置。

5.您可以定制 BSP。创建或打开一个.bsp 文件后，访问 **BSP Editor** 对话框中的 **Settings**。Settings 分成 Common 和 Advanced 设置。当您选择一组设置时，所选设置的控制出现在对话框的右侧。

当选择一个设置时，将显示设置名称，描述和值。您可以在 **BSP Editor** 对话框中编辑这些设置。

6.点击 **Generate** 生成 BSP。

7.点击 **Exit** 退出 BSP Generator GUI。

### 使用.tcl 脚本

除了使用默认设置，您也可以创建一个 tcl 脚本文件(**.tcl**)，定义 BSP 创建期间的定制设置。

`set_setting` 是唯一可用的**.tcl** 命令。

关于可用设置的列表，请参考 BSP Settings。

以下示例命令用于设置 BSP 设置文件中的参数：

```
set_setting spl.boot.BOOT_FROM_QSPI true
set_setting spl.boot.QSPI_NEXT_BOOT_IMAGE 0x50000
```

**相关链接**

BSP 设置 (第 7-9 页)

## BSP Generator 命令行界面

BSP 命令行工具可以从嵌入式命令壳调用，提供 BSP Generator GUI 中全部可用特性：

- **bsp-create-settings** 工具创建一个新的 BSP 设置文件。
- **bsp-update-settings** 工具更新一个现有的 BSP 设置文件。
- **bsp-query-settings** 工具报告一个现有 BSP 设置文件中的设置值。
- **bsp-generate-files** 工具从 BSP 设置文件生成一个 BSP。

**注意:** 每款工具的帮助信息(Help)都可从嵌入式命令壳获得。要显示帮助信息，请输入如下命令：`<name of tool> --help`。

### bsp-create-settings

**bsp-create-settings** 工具使用默认设置创建一个新的 BSP 设置文件。您可以选择修改 BSP 设置或者生成 BSP 文件，如下例中所示。

#### 实例 7-1: 创建一个新的 BSP 设置文件

下面实例根据硬件切换信息并使用默认的 BSP 设置创建了一个新的 `Preloader BSP` 设置文件：

```
bsp-create-settings --type spl --bsp-dir . \
 --settings settings.bsp \
 --preloader-settings-dir ../../hps_isw_handoff/<hps_entity_name>
```

**表 7-1: 用户参数：bsp-create-settings**

| 选项 | 是否需要 | 说明 |
|---|---|---|
| `--type <bsp type>` | 是 | 此选项指定 BSP 的类型。对于 Cyclone V/Arria V Preloader，所允许的值是"spl"，对于 Arria 10 ootloader，所允许的值是"uboot"。 |
| `--settings <filename>` | 是 | 此选项指定到 BSP 设置文件的路径。此文件使用默认设置创建。Altera 建议您命名 BSP 设置文件“settings.bsp”。 |
| `--preloader-settings-dir <directory>` | 是 | 此选项指定到硬件切换文件的路径。 |
| `--bsp-dir <directory>` | 是 | 此选项指定生成 BSP 文件的路径。指定此选项时，在设置文件已经创建后，**bsp -create-settings** 生成文件。Altera 建议您始终使用 **bsp -create-settings** 指定此参数。 |
| `--set <name> <value>` | 否 | 此选项将 BSP 设置<name>设置成值<value>。同一命令可以使用多个此选项的实例。关于设置名称和描述的列表，请参考 **BSP Settings**。 |

## bsp-update-settings

**bsp-update-settings** 工具更新存储在 BSP 设置文件中的设置，如下例所示。

### 实例 7-2: 更新一个 BSP 设置文件

下面命令修改"settings.bsp"文件中的一个参数值:

```
bsp-update-settings --settings settings.bsp --set spl.debug.SEMIHOSTING 1
```

**表 7-2: 用户参数：bsp-update-settings**

| 选项 | 是否需要 | 说明 |
|---|---|---|
| `--settings <settings-file>` | 是 | 此选项指定要更新的现有 BSP 设置文件的路径。 |

| 选项 | 是否需要 | 说明 |
|---|---|---|
| `--bsp-dir <bsp-dir>` | 否 | 此选项指定生成 BSP 文件的路径。指定此选项时，在设置文件已经创建后，**bsp -create-settings** 生成 BSP 文件。 |
| `--set <name> <value>` | 否 | 此选项将 BSP 设置<name>设置成值<value>。同一命令可以使用多个此选项的实例。关于设置名称和描述的列表，请参考 **BSP Settings** 。 |

## bsp-query-settings

**bsp-query-settings** 工具咨询存储在 BSP 设置文件中的设置，如下例所示。

### 实例 7-3: 咨询一个 BSP 设置文件

下面命令将从"settings.bsp"检索所有设置并显示设置名称和值：`bsp-query-settings --settings settings.bsp --get-all --show-names`

### 表 7-3: 用户参数：bsp-query-settings

| 选项 | 是否需要 | 说明 |
|---|---|---|
| `--settings <settings-file>` | 是 | 此选项指定一个现有 BSP 设置文件的路径。 |
| `--get <name>` | 否 | 此选项指示 **bsp -query-settings** 返回 BSP setting <name>的值。 |
| `--get-all` | 否 | 此选项显示所有的 BSP 设置值。使用--get-all 时，也必须要使用-- show-names。 |
| `--show-names` | 否 | 此选项与--get <name>或--get-all 一起使用时才有效。当与这些选项中的一个选项一起使用时，BSP 设置的名称和值将会并排显示。 |

## bsp-generate-files

**bsp-generate-files** 工具生成存储在 BSP 设置文件中的文件和设置，如下例所示。

**实例 7-4: BSP 创建后生成文件**

下面命令根据切换文件夹创建一个设置文件，然后根据这些设置生成 BSP 文件：

```
bsp-create-settings --type spl --bsp-dir . \
--settings settings.bsp \
--preloader-settings-dir \
../../hps_isw_handoff/<hps_entity_name>
bsp-generate-files --settings settings.bsp --bsp-dir
```

**实例 7-5: BSP 更新后生成文件**

下面命令更新一个现有 BSP 设置文件的设置，然后根据这些设置生成 BSP 文件：

```
bsp-update-settings --settings settings.bsp --set \
spl.debug.SEMIHOSTING 1
bsp-generate-files --settings settings.bsp --bsp-dir
```

当在下面其中一个条件下需要重新生成 BSP 文件时使用 **bsp-generate-files** 工具：

- **bsp-create-settings** 创建了 BSP 设置，但没有指定--bsp-dir 参数，所有没有生成 BSP 文件。
- **bsp-update-settings** 更新了 BSP 设置，但没有指定--bsp-dir 参数，所有没有更新文件。
- 您想确保 BSP 文件是最新的。

**表 7-4: 用户参数：bsp-generate-files**

| 选项 | 是否需要 | 说明 |
|---|---|---|
| `--settings <settings-file>` | 是 | 此选项指定一个现有 BSP 设置文件的路径。 |
| `--bsp-dir <bsp-dir>` | 是 | 此选项指定 BSP 文件生成的路径。 |

## BSP 文件和文件夹

BSP Generator 生成的文件和文件夹位于您在 **New BSP** 对话框中 **BSP target directory** 中指定的位置。

对于 Cyclone V/Arria Preloader BSP，生成的文件包括：

- **settings.bsp** – 包含所有 BSP 设置的文件
- **Makefile** – 用于编译 Preloader 和创建预加载器映像的 makefile；关于详细信息，请参考 Preloader Compilation。
- **preloader.ds** – ARM DS-5 脚本，用于在目标器件上下载并调试 Preloader。
- **generated** – 包含从硬件切换文件生成的文件的文件夹

对于 Arria 10 Bootloader BSP，生成的文件包括：

- **settings.bsp** – 包含所有 BSP 设置的文件
- **Makefile** – 用于编译 Bootloader，将 Bootloader 器件树文件转换成库文件，并生成合并的 Bootloader and Bootloader Device Tree 映像的 makefile；关于详细信息，请参考 Preloader Compilation。
- **config.mk** – makefile 配置文件，包含引导源选择和是否选择 Bootloader compilation。
- **devicetree.dts** – Bootloader 器件树，包括 Bootloader 定制详情，产生自切换文件 和用户设置。
- **uboot .ds** – ARM DS-5 脚本，用于在目标器件上下载并调试 Preloader。

## BSP 设置

这一部分列出了所有可用的 BSP 设置，可以从 Graphical User Interface 应用程序(**bsp-editor**)或者命令行工具(**bsp-create-settings**，**bsp-update-settings**，**bsp-query-settings**)访问这些 BSP 设置。

Cyclone V and Arria V SSBL (Preloader)与 Arria 10 SSBL (Bootloader)之间的可用 BSP 设置是不同的。

### Cyclone V 和 Arria V BSP 设置

**表 7-5: Cyclone V 和 Arria V BSP 设置**

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| spl.PRELOADER_TGZ | 字符串 | "<SoC EDS installation directory>/host_tools/altera/preloader/uboot-socfpga.tar.gz" | 此设置指定到包含预加载器源文件的存档文件的路径。 |
| spl.CROSS_COMPILE | String | "arm-altera-eabi-" | 此设置指定要使用的交叉编译工具链。 |
| spl.boot.BOOT_FROM_QSPI | Boolean | False | 选择后续引导映像的来源。请注意同一时间只有一个源可以是活动的。当使用 **bsp-create-settings** 或 **bsp-update-settings** 时，您必须先关闭当前打开的引导选项后才能打开其他引导选项。 |
| spl.boot.BOOT_FROM_SDMMC | Boolean | True | |
| spl.boot.BOOT_FROM_RAM | Boolean | False | |
| spl.boot.BOOT_FROM_NAND | Boolean | False | |
| spl.boot.QSPI_NEXT_BOOT_IMAGE | Hexadecimal | 0x60000 | 此设置指定 QSPI 中后续引导映像的位置。 |
| spl.boot.SDMMC_NEXT_BOOT_IMAGE | Hexadecimal | 0x40000 | 此设置指定 SD/MMC 中后续引导映像的位置。 |
| spl.boot.NAND_NEXT_BOOT_IMAGE | Hexadecimal | 0xC0000 | 此设置指定 NAND 中后续引导映像的位置。 |

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| spl.boot.FAT_SUPPORT | Boolean | False | 当从 SD/MMC 引导时使能 FAT 分区支持。 |
| spl.boot.FAT_BOOT_PARTITION | DecimalNumber | 1 | 当 FAT 分区支持使能时，指定引导映像所位于的 FAT 分区。 |
| spl.boot.FAT_LOAD_PAYLOAD_NAME | String | "u-boot.img" | 当 FAT 分区支持使能时，指定要使用的引导映像文件名。 |
| spl.boot.WATCHDOG_ENABLE | Boolean | True | 此设置使能预加载执行阶段的看门狗功能。预加载退出后看门狗功能保持使能。 |
| spl.boot.CHECKSUM_NEXT_IMAGE | Boolean | True | 此设置使预加载程序验证后续引导映像头信息中的校验和。 |
| spl.boot.EXE_ON_FPGA | Boolean | False | 此设置在 FPGA 上运行预加载程序。当预加载程序配置成从 FPGA 引导时选择 spl.boot.EXE_ON_FPGA。 |
| spl.boot.STATE_REG_ENABLE | Boolean | True | 此设置使预加载程序退出时将魔法值(magic value)写入到系统管理器中的 **INITSWSTATE** 寄存器中；指示 boot ROM 预引导程序已经成功运行。 |
| spl.boot.BOOTROM_HANDSHAKE_CFGIO | Boolean | True | 此设置使能配置 IOCSR 和管脚多路复用时与 boot ROM 的握手(handshake)。如果 spl.boot.BOOTROM_HANDSHAKE_CFGIO 使能，并且当预加载程序配置 IOCSR 和管脚多路复用时出现热复位(warm reset)，那么 boot ROM 将再次重配置 IOCSR 和管脚多路复用。此选项默认情况下是使能的。 |
| spl.boot.WARMRST_SKIP_CFGIO | Boolean | True | 此设置使预加载程序在热复位期间跳过 IOCSR 和管脚多路复用。仅在 boot ROM 已经跳过 IOCSR 和管脚多路复用配置时 spl.boot.WARMRST_SKIP_CFGIO 才可用。 |
| spl.boot.SDRAM_INITIALIZATION | Boolean | False | 初始化 SDRAM 以初始化 ECC 比特。 |

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| spl.boot.SDRAM_ECC_INIT_BOOT_REGION_START | Hexadecimal | 0x1000000 | 要初始化的 SDRAM 中的存储器区域的起始地址。 |
| spl.boot.SDRAM_ECC_INIT_BOOT_REGION_END | Hexadecimal | 0x2000000 | 要初始化的 SDRAM 中的存储器区域的结束地址。 |
| spl.boot.SDRAM_ECC_INIT_REMAIN_REGION | Boolean | True | 闪存访问期间初始化剩余的 SDRAM 以加载映像。 |
| spl.debug.DEBUG_MEMORY_WRITE | Boolean | False | 此设置使能预加载程序将调试信息写入到存储器中用于调试，当 UART 不可用时此设置是有用的。地址由 spl.debug.DEBUG_MEMORY_ADDR 指定。 |
| spl.debug.SEMIHOSTING | Boolean | False | 此设置使能预加载程序中的半主机(semihosting)支持，与调试工具一起使用。当 UART 不可用时，spl.debug.SEMIHOSTING 是有用的。 |
| spl.debug.SKIP_SDRAM | Boolean | False | 当此设置使能时，预加载程序跳过 SDRAM 初始化和校准。 |
| spl.performance.SERIAL_SUPPORT | Boolean | True | 此设置使能 UART 打印支持，使预加载程序代码在运行时通过调试信息调用 printf()。printf() 的 stdout 输出被导向到 UART。通过将一个终端程序连接到 UART 指定的外设来查看此调试信息。 |

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| spl.reset_assert.DMA | Boolean | False | 使能时，此设置将强制相应的外设处于复位状态。您必须确保调试器不读取这些组件的寄存器。 |
| spl.reset_assert.GPIO0 | Boolean | False | |
| spl.reset_assert.GPIO1 | Boolean | False | |
| spl.reset_assert.GPIO2 | Boolean | False | |
| spl.reset_assert.L4WD1 | Boolean | False | |
| spl.reset_assert.OSC1TIMER1 | Boolean | False | |
| spl.reset_assert.SDR | Boolean | False | |
| spl.reset_assert.SPTIMER0 | Boolean | False | |
| spl.reset_assert.SPTIMER1 | Boolean | False | |
| spl.warm_reset_handshake.FPGA | Boolean | True | 此设置使复位管理器在置位热复位前执行与 FPGA 的握手 (handshake)。 |
| spl.warm_reset_handshake.ETR | Boolean | True | 此设置使复位管理器能够请求 Embedded Trace Router (ETR)中止 Advanced eXtensible Interface (AXI) master，并在置位 L3 互联的热复位或者 ETR 的调试复位之前等待 ETR 完成任何未完成的 AXI 传输。 |
| spl.warm_reset_handshake.SDRAM | Boolean | False | 此选项允许在热复位时保留 SDRAM 内容。<br>**注意:** 当使用此选项时，SDRAM 控制器从热复位释放后没有完全重新初始化。作为 SDRAM 控制器失败的后果，只要 Watchdog 生成热复位，这就可能是一个问题。<br>**注意:** 此外，当此选项使能时，并且系统从热复位释放时，SDRAM PLL 没有重新初始化。 |

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| spl.boot.FPGA_MAX_SIZE | Hexadecimal | 0x10000 | 此设置指定能够适合 FPGA 的最大代码(**.text** 和**.rodata**)大小。如果代码构建(code build)大于指定的大小，那么会触发构建错误(build error)。 |
| spl.boot.FPGA_DATA_BASE | Hexadecimal | 0xFFFF0000 | 当 execute on FPGA 使能时，此设置指定数据区域的基本位置(**.data**、**.bss**、**heap** 和 **stack**)。 |
| spl.boot.FPGA_DATA_MAX_SIZE | Hexadecimal | 0x10000 | 此设置指定能够适合 FPGA 的最大数据(**.data**、**.bss**、**heap** 和 **stack**)大小。如果代码构建(code build)大于指定的大小，那么会触发构建错误(build error)。 |
| spl.debug.DEBUG_MEMORY_ADDR | Hexadecimal | 0xFFFFFD00 | 此设置指定用于存储通过 spl.debug.DEBUG_MEMORY_WRITE 设置使能的预加载程序调试信息的基本地址。 |
| spl.debug.DEBUG_MEMORY_SIZE | Hexadecimal | 0x200 | 此设置指定用于存储预加载程序调试信息的最大容量。 |
| spl.debug.DEBUG_MEMORY_ADDR | Hexadecimal | 0xFFFFFD00 | 此设置指定用于存储通过 spl.debug.DEBUG_MEMORY_WRITE 设置使能的预加载程序调试信息的基本地址。 |
| spl.debug.HARDWARE_DIAGNOSTIC | Boolean | False | 使能硬件诊断支持。要使能此选项，需要至少 1GB 容量的存储器，否则硬件诊断将无法正常运行。 |
| spl.boot.RAMBOOT_PLLRESET | Boolean | True | 当 CSEL = 00 时在热复位上执行 RAM Boot PLL 复位代码。当使用 CSEL = 00 时需要此选项使能正确的热复位功能。当使能此选项时，高 4 KB 的 OCRAM 的被保留，并且一定不要被用户软件修改。<br>**注意:** 使能此功能及 CSEL != 00 将没有任何作用，因为受影响的代码会检查这一点。 |

## Arria 10 BSP 设置

Arria 10 BSP 设置分为四组：

- Main Group
- MPU Firewall
- L3 Firewall Group
- F2S Firewall Group

下表列出了每组的设置名称前添加的设置前缀，使本节的其余部分更具可读性。

**表 7-6: Arria 10 BSP 防火墙设置前缀**

| 设置组 | 设置组前缀 |
|---|---|
| Main Group | N/A |
| MPU Firewall | `mpu_m0.noc_fw_ddr_mpu_fpga2sdram_ddr_scr.*` |
| L3 Firewall Group | `mpu_m0.noc_fw_ddr_l3_ddr_scr.*` |
| F2S Firewall Group | `mpu_m0.noc_fw_ddr_mpu_fpga2sdram_ddr_scr.*` |

### Arria 10 主要的 BSP 设置组

**表 7-7: Arria 10 主要的 BSP 设置组**

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| `uboot.boot_device` | String | `"SD/MMC"` | 选择引导映像的源。可能值为 SD/MMC 和 QSPI。 |
| `uboot.model` | String | `"SOCFPGA Arria10 Dev Kit"` | 当引导加载程序启动时要显示的电路板名称。 |
| `uboot.external_fpga_config` | Boolean | False | 通过 JTAG 下载或外部连接的闪存配置 Uboot 在引导序列中提前等待， 使 FPGA 进入用户模式。 |
| `uboot.rbf_filename` | String | `socfpga.rbf` | 完整的 FPGA `.rbf` 文件名。当 `uboot.external_fpga_config` 设置使能时忽略此设置。 |
| `uboot.rbf_offset` | Hexadecimal | `0x720000` | QSPI Flash 中的 RBF 偏移地址。 |
| `uboot.disable_uboot_build` | Boolean | False | 能用于禁止 building Uboot，仅生成 Bootloader Device Tree 文件。 |
| `uboot.secureboot.enable_bootloader_encryption` | Boolean | 0 | 使用指定的密钥文件加密 Bootloader。 |
| `uboot.secureboot.enable_bootloader_signing` | Boolean | 0 | 使用指定的密钥对文件签署 Bootloader。 |
| `uboot.secureboot.encryption_key_file` | String | `encrypt.key` | 用于 Bootloader 加密的密钥文件。 |

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| uboot.secureboot.encryption_key_name | String | key1 | Bootloader 加密的加密文件中使用的密钥名称。 |
| uboot.secureboot.signing_key_fpga_offset | Hexadecimal | 0x0 | 从 H2F Bridge Base Address (0xC0000000)到 root-public-key 位置的偏移。 |
| uboot.secureboot.signing_key_pair_file | String | root_key.pem | signing 使能时使用的密钥对文件。您能够使用'make generate-signing-key-pair-file'命令生成此文件。 |
| uboot.secureboot.signing_key_type | String | user | 使用指定的密钥对文件签署 Bootloader。 |

### Arria 10 引导加载程序 MPU 防火墙 BSP 设置组

**表 7-8: Arria 10 引导加载程序 MPU 防火墙 BSP 设置组**

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| enable.mpuregion0enable | Boolean | True | 使能 MPU 防火墙区域 |
| enable.mpuregion1enable | | False | |
| enable.mpuregion2enable | | False | |
| enable.mpuregion3enable | | False | |
| mpuregion0addr.base | Hexadecimal | 0x0 | MPU 防火墙区域基础 |
| mpuregion1addr.base | | | |
| mpuregion2addr.base | | | |
| mpuregion3addr.base | | | |
| mpuregion0addr.limit | Hexadecimal | 0xffff | MPU 防火墙区域限制 |
| mpuregion1addr.limit | | | |
| mpuregion2addr.limit | | | |
| mpuregion3addr.limit | | | |

## Arria 10 引导加载程序 L3 防火墙 BSP 设置组

**表 7-9: Arria 10 引导加载程序 L3 防火墙 BSP 设置组**

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| enable.hpsregion0enable | Boolean | True | 使能 L3 防火墙区域 |
| enable.hpsregion1enable | | False | |
| enable.hpsregion2enable | | False | |
| enable.hpsregion3enable | | False | |
| enable.hpsregion4enable | | False | |
| enable.hpsregion5enable | | False | |
| enable.hpsregion6enable | | False | |
| enable.hpsregion7enable | | False | |
| hpsregion0addr.base | Hexadecimal | 0x0 | L3 防火墙区域基础 |
| hpsregion1addr.base | | | |
| hpsregion2addr.base | | | |
| hpsregion3addr.base | | | |
| hpsregion4addr.base | | | |
| hpsregion5addr.base | | | |
| hpsregion6addr.base | | | |
| hpsregion7addr.base | | | |
| hpsregion0addr.limit | Hexadecimal | 0xffff | L3 防火墙区域限制 |
| hpsregion1addr.limit | | | |
| hpsregion2addr.limit | | | |
| hpsregion3addr.limit | | | |
| hpsregion4addr.limit | | | |
| hpsregion5addr.limit | | | |
| hpsregion6addr.limit | | | |
| hpsregion7addr.limit | | | |

## Arria 10 引导加载程序 F2S 防火墙 BSP 设置组

**表 7-10: Arria 10 引导加载程序 F2S 防火墙 BSP 设置组**

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| enable.fpga2sdram0region0 | Boolean | True | 使能 F2S 防火墙区域 |
| enable.fpga2sdram0region1 | | True | |
| enable.fpga2sdram0region2 | | True | |
| enable.fpga2sdram0region3 | | False | |
| enable.fpga2sdram1region0 | | False | |
| enable.fpga2sdram1region1 | | False | |
| enable.fpga2sdram1region2 | | False | |
| enable.fpga2sdram1region3 | | False | |
| enable.fpga2sdram2region0 | | False | |
| enable.fpga2sdram2region1 | | False | |
| enable.fpga2sdram2region2 | | False | |
| enable.fpga2sdram2region3 | | False | |
| fpga2sdram0region0addr.base | Hexadecimal | 0x0 | F2S 防火墙区域基础 |
| fpga2sdram0region1addr.base | | | |
| fpga2sdram0region2addr.base | | | |
| fpga2sdram0region3addr.base | | | |
| fpga2sdram1region0addr.base | | | |
| fpga2sdram1region1addr.base | | | |
| fpga2sdram1region2addr.base | | | |
| fpga2sdram1region3addr.base | | | |
| fpga2sdram2region0addr.base | | | |
| fpga2sdram2region1addr.base | | | |
| fpga2sdram2region2addr.base | | | |
| fpga2sdram2region3addr.base | | | |

| BSP 设置 | 类型 | 默认值 | 说明 |
|---|---|---|---|
| fpga2sdram0region0addr.limit | Hexadecimal | 0xffff | F2S 防火墙区域限制 |
| fpga2sdram0region1addr.limit | | | |
| fpga2sdram0region2addr.limit | | | |
| fpga2sdram0region3addr.limit | | | |
| fpga2sdram1region0addr.limit | | | |
| fpga2sdram1region1addr.limit | | | |
| fpga2sdram1region2addr.limit | | | |
| fpga2sdram1region3addr.limit | | | |
| fpga2sdram2region0addr.limit | | | |
| fpga2sdram2region1addr.limit | | | |
| fpga2sdram2region2addr.limit | | | |
| fpga2sdram2region3addr.limit | | | |

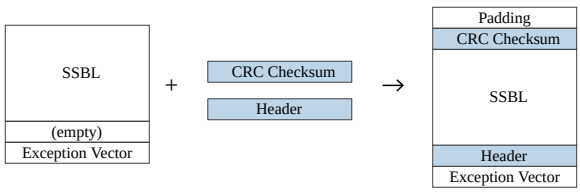
# 第二阶段引导加载程序映像工具(mkpimage)

第二阶段引导加载程序映像工具(Second Stage Bootloader (SSBL) Image Tool (mkpimage))创建一个 Arria V and Cyclone V Preloader 或者 Arria 10 Bootloader 的 Altera BootROM 兼容的映像。此工具还可以对之前生成的映像的头(header)进行解码。

mkpimage 工具作如下假设：

- 输入文件格式为原始二进制。您必须使用 Mentor Graphics 网站上的 GNU Compiler Collection (GCC)工具链提供的 **objcopy** 工具将诸如 Executable and Linking Format File (**.elf**)，Hexadecimal (Intel-Format) File (**.hex**)或者 S-Record File (**. srec**)的其他文件格式转换成二进制格式。
- 输出文件格式为二进制。
- 此工具总是在二进制文件的开始创建输出映像。如果必须在特定的基地址上对映像编程，那么您必须提供地址信息给闪存编程工具。
- 输出文件仅包含 Preloader 或 Bootloader 映像。对于诸如 Linux，SRAM Object File (.sof)和用户数据的其他映像，使用闪存编程工具或者目标系统上的 U-boot 中的相关工具分别对它们进行编程。

**图 7-5: mkpimage 工具的基本操作**



## 操作

mkpimage 工具在主机上运行。此工具生成头(header)和 CRC 校验和，并将它们插入到具有 SSBL 程序映像及其异常向量的输出映像中。

对于某些闪存工具，SSBL 映像的位置必须符合指定的模块大小；mkpimage 工具生成任何所需要的填充数据(padding data)。

mkpimage 工具对于一个给定的预生成的 SSBL 映像视情况对头信息进行解码和验证。

二进制 SSBL 映像是 mkpimage 工具的一个输入。编译器在 SSBL 异常向量和程序之间留有一个空白区域。mkpimage 管脚使用头信息覆盖这一空白区域，并计算整个映像的校验和。

必要时，mkpimage 工具会将填充数据附加到输出映像。

mkpimage 工具能够操作一个或四个输入文件。四个输入文件的操作是先单独处理每个文件，然后将四个处理过的文件合并起来。

## 头文件格式(Header File Format)

mkpimage 头文件格式有两个版本:

- Version 0，用于 Cyclone V and Arria V SSBL (Preloader)
- Version 1，用于 Arria 10 SSBL (Bootloader)

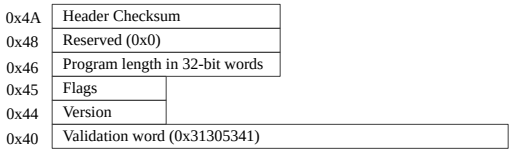
对于用于 Cyclone V and Arria V Preloader 的 Version 0，头(header)包括下面部分:

- 验证字 (0x31305341)
- 版本字段 (设为 0x0)
- 标志字段 (设为 0x0)
- 程序长度，测量为 Preloader 程序中 32 bit 字的数量
- 头内容的 16-bit 校验和(0x40 – 0x49)

**表 7-11: 头格式版本 0**

| 0x4A | 头校验和 |
|---|---|
| 0x48 | 保留(0x0) |
| 0x46 | 以 32-bit 字表示的程序长度 (n) |
| 0x45 | 标志 |
| 0x44 | 版本 |
| 0x40 | 验证字 (0x31305341) |

**图 7-6: 头格式版本 0**




对于用于 Arria 10 Bootloader 的 Version 1，头(header)包括下面部分：

- 验证字 (0x31305341)
- 版本字段 (设为 0x0)
- 标志字段 (设为 0x0)
- 头长度，单位为字节，设为 0x14 (20 字节)。
- 以字节为单位的总程序长度(包括异常向量和 CRC 字段)。对于一个有效的映像，长度必须最小为 0x5C (92 字节)，最大为 0x32000 (200KiB)。
- Program entry offset relative 相对于头开始(0x40)的程序入口偏移应该是 32-bit 字对齐的。默认值为 0x14，小于此值都是无效的。
- 头内容的 16-bit 校验和(0x40 – 0x51)：

**图 7-7: 头格式版本 1**

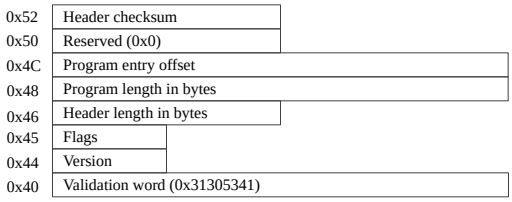



**表 7-12: 头格式版本 1**

| 0x52 | 头校验和 |
|---|---|
| 0x50 | 保留(0x0) |
| 0x4c | 程序入口偏移(x) |
| 0x48 | 程序长度，以字节为单位(n) |
| 0x46 | 头长度，以字节为单位 |
| 0x45 | 标志 |
| 0x44 | 版本 |
| 0x40 | 验证字 (0x31305341) |

mkpimage 头的两个版本的头校验和都是从 offset 0x0 到(n*4)-4 字节的字节值的 CRC 校验和，其中 n 是程序长度。

CRC 是一个使用下面多项式的标准 CRC32：

```
x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1
```

。没有比特反映，其余的初始值是 0xFFFFFFFF，最终值被 0xFFFFFFFF 异或(exclusive OR)。

## 工具使用

mkimage 工具有三种使用模型：

- 单映像创建(Single image creation)
- 四映像创建(Quad image creation)
- 单映像或四映像解码(Single or quad image decoding)

如果在映像处理过程中发现错误，那么工具会停止并报告错误。导致错误的可能情况如下：

- 对于 Cyclone V and Arria V Preloaders：输入映像小于 81 字节或者大于 60 KB。
- 对于 Arria 10 Bootloaders：输入映像小于 92 字节或者大于 200 KB。

**mkpimage** 调用工具；使用--help 选项来调用工具会提供工具描述和工具使用和选项选项。

```
$ mkpimage --help
mkpimage version 15.1 (build 189)

Description: This tool creates an Altera BootROM-compatible image of Second
Stage Boot Loader (SSBL). The input and output files are in binary format.
It can also decode and check the validity of previously generated image.

Usage:
 Create quad image:
     mkpimage [options] -hv <num> -o <outfile> <infile> <infile> <infile> <infile>
 Create single image:
     mkpimage [options] -hv <num> -o <outfile> <infile>
 Decode single/quad image:
     mkpimage -d [-a <num>] <infile>

Options:
 -a (--alignment) <num>       : Address alignment in kilobytes for output image
                                (64, 128, 256, etc.), default to 64 for header
                                version 0 and 256 for header version 1,
                                override if the NAND flash has a different
                                block size. If outputting a single image, value
                                of '0' is permitted to specify no flash block
                                padding (needed for SSBL image encryption).
 -d (--decode)                : Flag to decode the header information from
                                input file and display it
 -f (--force)                 : Flag to force decoding even if the input file
                                is an unpadded image
 -h (--help)                  : Display this help message and exit
 -hv (--header-version) <num> : Header version to be created (Arria/Cyclone V =
                                0, Arria 10 = 1)
 -o (--output) <outfile>      : Output file, relative and absolute path
                                supported
 -off (--offset) <num>        : Program entry offset relative to start of
                                header (0x40), default to 0x14. Used for header
                                version 1 only
 -v (--version)               : Display version and exit
```

## 输出映像布局

## 基地址

对于 NAND 和 QSPI 闪存，可引导的 SSBL 映像必须放置在 0x0 上。对于 SD/MMC，映像也可放置在 0x0 上，但通常映像被放置在定制分区类型 0xA2 中的 offset 0x0 上。定制分区上没有文件系统。boot ROM 通过使用位于 SD/MMC 卡的 0x0 上的 MBR (Master Boot Record)找到分区。

无论目标闪存类型如何，mkpimage 工具总是将输出映像放置在输出二进制文件的开始部分。闪存编程工具负责将映像放置在闪存器件上的相应位置。

## 大小

对于 Cyclone V 和 Arria V，单一 Preloader 的最大映像尺寸是 60 KB。您可以在闪存中最多存储四个 preloader 映像。如果 boot ROM 在第一个位置没有找到一个有效的 preloader 映像，那么它会尝试从下一个位置读取一个映像。要运用这一特性，需要编程闪存中的四个 preloader 映像。

对于 Arria 10，单一 Bootloader 的最大映像尺寸是 200 KB。您可以在闪存中最多存储四个 Bootloader 映像。如果 boot ROM 在第一个位置没有找到一个有效的 Bootloader 映像，那么它会尝试从下一个位置读取一个映像。要运用这一特性，需要编程闪存中的四个 Bootloader 映像。

## 地址对齐

对于 Cyclone V 和 Arria V，每个 Preloader 映像都要对齐于 64KB 边界，除了 NAND 器件。对于 NAND 器件，每个 Preloader 映像都要对齐于大于 64 KB 或 NAND 模块大小。

对于 Arria 10，每个 Bootloader 映像都要对齐于 256 KB 边界，除了 NAND 器件。对于 NAND 器件，每个 Preloader 映像都要对齐于大于 256 KB 或 NAND 模块大小。

下表列出了典型的映像布局，用于模块大小等于或少于 64 KB (Cyclone V/Arria V)或者 256 KB (Arria 10)的 QSPI、SD/MMC 和 NAND 器件。

**表 7-13: 典型的 Arria V/Cyclone V 预加载程序映像布局**

| 偏移 | 映像 |
|---|---|
| 0x30000 | Preloader Image 3 |
| 0x20000 | Preloader Image 2 |
| 0x10000 | Preloader Image 1 |
| 0x00000 | Preloader Image 0 |

**表 7-14: 典型的 Arria 10 引导加载程序映像布局**

| 偏移 | 映像 |
|---|---|
| 0xC0000 | Bootloader Image 3 |
| 0x80000 | Bootloader Image 2 |
| 0x40000 | Bootloader Image 1 |
| 0x00000 | Bootloader Image 0 |

mkpimage 工具不识别目标闪存类型。若不指定模块大小，则默认值是 64 KB。

### NAND Flash

每个 Preloader 或 Bootloader 映像占用整数数量的模块。一个模块是可擦除的最小实体，所有对一个特定引导映像的更新不会影响到其他映像。

例如，对于 Cyclone V 和 Arria V，单一 Preloader 映像的最大尺寸为 64 KB。但如果 NAND 模块尺寸为 128 KB，那么 Preloader 映像需要位于 128 KB 间隔。

### 串行 NOR Flash

每个 QSPI 引导映像占用整数数量的扇区(除非支持子扇区)；这就确保了对一个映像的更新不会影响到其他映像。

### SD/MMC

主引导记录位于器件存储器的头 512 个字节，包含分区地址和大小信息。Preloader 和 Bootloader 映像存储在 0xA2 类型的分区中。其他部分可以根据目标文件系统格式存储在其他类型的分区中。

您可以使用 fdisk 工具设置和管理主引导记录。当 fdisk 工具对一个 SD/MMC 器件进行分区时，此工具在第一个扇区创建主引导记录，以及 SD/MMC 中每个分区的分区地址和大小信息。

## 填充(Padding)

mkimage 工具将一个 CRC 校验和插入到映像中未使用的区域。 Padding 填充剩下未使用的区域。映像的已填充区域和未使用区域是未定义的。

**相关链接**

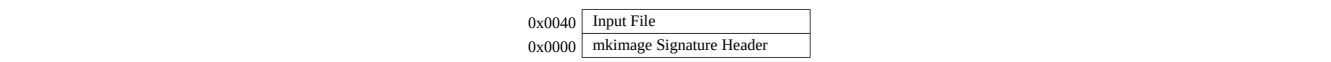
- **Arria V 硬核处理器系统技术参考手册**
  关于详细信息，请参考"引导和配置"章节。
- **Cyclone V 硬核处理器系统技术参考手册**
  关于详细信息，请参考"引导和配置"章节。
- **Arria 10 硬核处理器系统技术参考手册**
  关于详细信息，请参考"引导和配置"章节。

# U-Boot 映像工具(mkimage)

Preloader (for Arria V/ Cyclone V)以及 Bootloader (for Arria 10)都要求在下一阶段引导映像的开始存在 U-Boot 映像头。根据不同的使用情况，由 Preloader 或 Bootloader 加载的其他部分可能也需要 U-Boot 映像头的存在。

mkimage 工具 SoC EDS 的一部分，能够用于将 U-Boot 映像头附加到下一阶段引导映像或任何其他所需文件上。

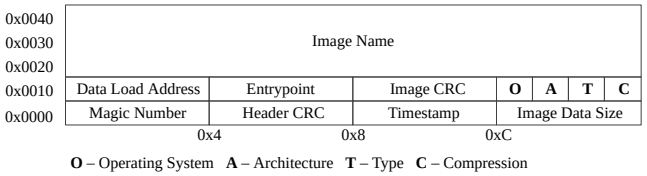
**图 7-8: mkimage Header**




header 由下面部分组成：

- Image magic number - 决定映像是否是一个有效的引导映像
- Image data size - 要复制的映像长度
- Data load address - 引导映像的入口点(不用于那些不可引导的映像)
- Operating system - 决定映像的类型
- Image name - 引导映像的名称
- Image CRC - 引导映像的校验和值

**图 7-9: mkimage Header 布局**




## 工具选项

**mkimage** 调用 mkimage 工具，--help 选项提供工具说明和选项信息。

```
$ mkimage --help
Usage: mkimage -l image
          -l ==> list image header information
       mkimage [-x] -A arch -O os -T type -C comp -a addr -e ep -n name -d
data_file[:data_file...] image
          -A ==> set architecture to 'arch'
          -O ==> set operating system to 'os'
          -T ==> set image type to 'type'
          -C ==> set compression type 'comp'
          -a ==> set load address to 'addr' (hex)
          -e ==> set entry point to 'ep' (hex)
          -n ==> set image name to 'name'
          -d ==> use image data from 'datafile'
          -x ==> set XIP (execute in place)
       mkimage [-D dtc_options] [-f fit-image.its|-F] fit-image
          -D => set options for device tree compiler
          -f => input filename for FIT source
Signing / verified boot not supported (CONFIG_FIT_SIGNATURE undefined)
       mkimage -V ==> print version information and exit
```

## 使用实例

### 实例 7-6: 创建一个 U-boot 映像

```
mkimage -A arm -T firmware -C none -O u-boot -a 0x08000040 -e 0 -n "U-
Boot 2014.10 for SOCFGPA board" -d u-boot.bin u-boot.img
```

### 实例 7-7: 创建一个裸机应用映像

```
mkimage -A arm -O u-boot -T standalone -C none -a 0x02100000 -e 0 -n
"baremetal image" -d hello_world.bin hello_world.img
```

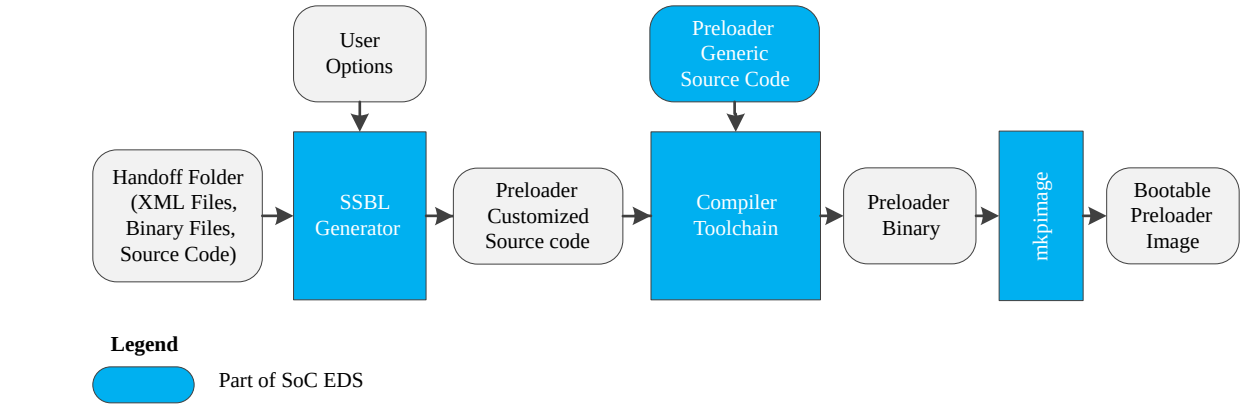
# 构建一个第二阶段引导加载程序

本节详细阐述了如何对 Cyclone V and Arria V Preloader 以及 Arria 10 Bootloader 通过生成的 Makefile 来创建可引导的 SSBL 映像。

## 构建 Cyclone V 和 Arria V Preloader

下图显示了当调用 **make** 时由生成的 Makefile 执行的 Cyclone V 和 Arria V Preloader 构建流程。为使图示简单清晰，未显示生成的 Makefile。

**图 7-10: Cyclone V/Arria V Preloader 构建流程**




makefile 执行以下任务：

- 将通用预加载程序源代码复制到**< bsp directory>/ uboot-socfpga**
- 将生成的 BSP 文件和硬件切换文件复制到**< bsp_directory >/ uboot-socfpga /board/ altera / socfpga _<device>**的源目录中
- 配置编译器工具以使用目标 SoC FPGA
- 使用用户指定的交叉编译器(在 BSP 设置中指定)编译**< bsp directory>/ uboot-socfpga** 中的源文件，并将生成的预加载程序二进制文件存储到**< bsp_directory >/ uboot - socfpga / spl** 中
- 使用 **mkpimage** 工具将预加载程序二进制文件转换成预加载程序映像，**< bsp_directory >/ preloader- mkpimage.bin**

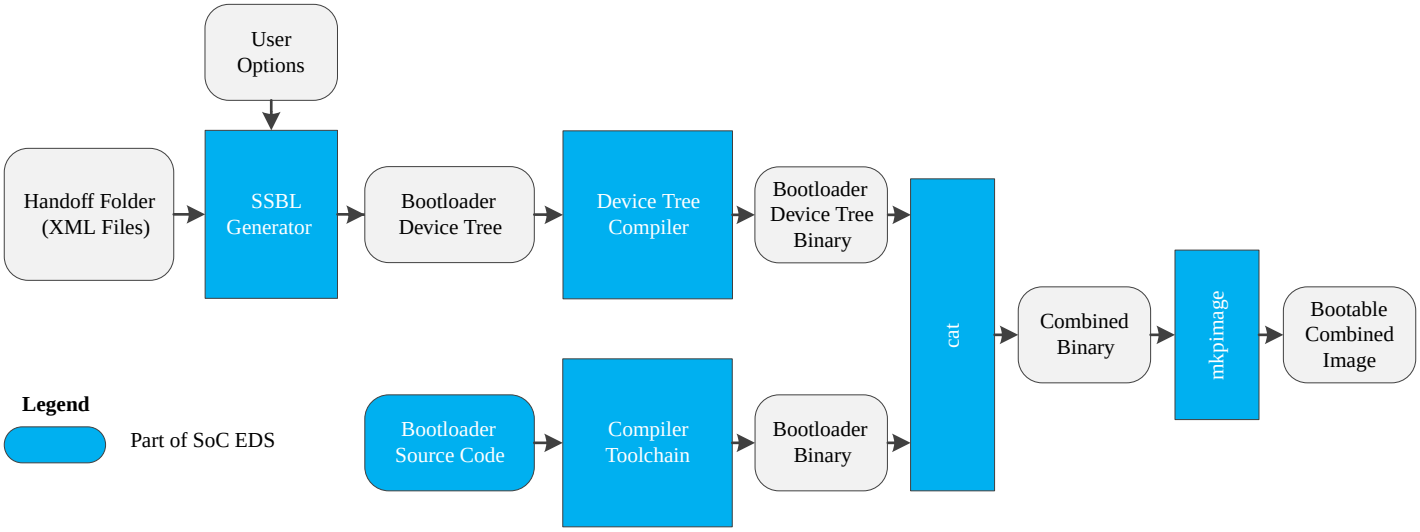
makefile 包含下面部分：

- make all – 编译预加载程序
- make clean – 将 preloader-mkpimage.bin 从<bsp directory>中删除
- make clean-all – 删除<bsp directory>，包括目录中的源文件

## 构建 Arria 10 Bootloader

下图显示了当调用 **make** 时由生成的 Makefile 执行的 Arria 10 Bootloader 构建流程。为使图示简单清晰，未显示生成的 Makefile。

**图 7-11: Arria 10 Bootloader 构建流程**




**警告**：对于此版本的工具，仅在 Linux 主机上支持 Bootloader 编译。在 Windows 主机上不支持 Bootloader 编译。在 Linux 和 Windows 主机上都支持此流程的剩余部分。

**注意:** Bootloader 仅在引导源改变时才需要重新编译(SD/MMC 和 QSPI 当前是被支持的)。当其他设置改变时，Bootloader 不需要重新编译，因为这些设置都封装在 Bootloader Device Tree 中。

makefile 执行以下任务：

- 将引导加载程序源代码复制到**< bsp directory>/ uboot-socfpga**
- 配置引导加载程序以使用目标 Arria 10 SoC
- 编译**< bsp directory>/ uboot-socfpga** 中的源文件，并创建 Bootloader 二进制**< bsp directory>/ uboot-socfpga /u- boot .bin**
- 根据硬件切换信息和用户设置创建引导加载程序器件树文件到**< bsp directory>/ devicetree.dts**
- 通过使用 **dtc** (Device Tree Compiler)编译引导加载程序器件树源文件到**< bsp directory>/ devicetree.dtb**
- 将引导加载程序器件树二进制文件和引导加载程序二进制文件合并成一个二进制文件**< bsp directory>/ u- boot_w_dtb.bin**
- 使用 **mkpimage** 工具将合并的二进制文件转换成一个可引导的合并映像，**< bsp directory>/ uboot_w_dtb-mkpimage.bin**

makefile 包含以下部分：

- make all – 构建一切
- make src – 复制引导加载程序源代码文件夹
- make uboot – 构建引导加载程序二进制
- make dtb – 编译引导加载程序器件树到器件树二进制
- make clean – 删除所有构建文件
- make clean-all – 删除所有构建文件和引导加载程序源代码文件夹

# 引导工具用户指南文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | • 更新了"工具使用"和"工具选项"部分中的 `mkpimage` 的帮助定义。<br>• 更新了 Arria 10 主要 BSP 设置组表。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

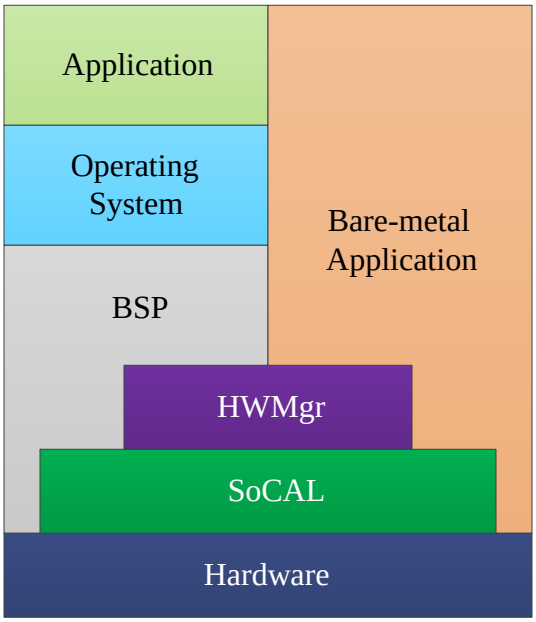
# 硬件库 8

2016.02.17

ug-1137 订阅 反馈

Altera SoC FPGA Hardware Library (HWLIB)的创建是为了解决那些需要对 SoC FPGA 硬件的配置和控制部分进行完全访问的初级软件程序员的需要。HWLIB 的另一个目的是降低管理一个成熟的，多核应用处理器的操作的复杂性，以及在 SoC 体系结构中与硬核 IP 外设模块和可编程逻辑集成的复杂性。

**图 8-1: HW Library**




在 SoC HW/SW 生态系统环境中，通过与全功能的操作系统或独立的裸机编程环境相结合，HWLIB 能够支持软件开发。上图中显示了一个完整 SoC HW/SW 环境中 HWLIB 关系。

HWLIB 提供一种称为 SoC Abstraction Layer (SoCAL)的符号寄存器抽象层，实现了在地址空间范围内对 HPS 器件寄存器的直接访问和控制。此符号寄存器抽象层对于要求精确程度的硬件资源的访问和控制的关键利益相关者(引导加载程序开发人员，驱动程序开发人员，电路板支持封装的开发，调试代理开发人员和电路板启动工程师)而言是必要的。

HWLIB 也部署一套 Hardware Manager (HW Manager) API，对更高级别的使用情况提供更复杂的功能和驱动程序。

HWLIB 是作为源代码分布进行开发的。此模型旨在提供一套实用的开箱即用功能，并且用作源代码参考实现，用户可以相应地调整源代码参考实现，以满足他们的目标系统的要求。

HWLIB 的性能随着时间的推移将会不断地增强和扩展，主要是因为常见的使用案例模式已出现在现实系统中的实际应用中。

一般情况下，HWLIB 假定为在 Hard Processor System (HPS)上，在特权管理员模式下和安全状态下运行的系统软件的一部分。

预期的 HWLIB 用户包括：

- 裸机应用程序开发人员
- 自定义预加载和引导加载程序软件开发人员
- 电路板支持封装开发人员
- 诊断工具开发人员
- 软件驱动程序开发人员
- 调试代理开发人员
- 电路板启动工程师
- 需要对 SoC FPGA 硬件功能进行完全访问的其他开发人员

# 功能描述

本节对 HWLIB 具有的操作特性和功能进行了描述，也进行了 HWLIB 体系结构的概述和简要描述。

HWLIB 是一个软件库，在体系结构上由两个主要的功能组件构成：

- SoC Abstraction Layer (SoCAL)（SoC 抽象层）
- Hardware Manager (HW Manager)（硬件管理器）

## SoC Abstraction Layer (SoCAL)

SoC Abstraction Layer (SoCAL)表示最接近实际 HPS 硬件的软件 API。其目的是对组成 HPS 的硬件接口的物理器件和寄存器提供一个逻辑接口抽象和去耦层。

SoCAL 的优势如下：

- 一个 HPS 物理器件和寄存器的逻辑接口抽象，包含组成它们的比特字段。
- 一个底层硬件的松散耦合的软件接口，促进软件隔离于系统地址映射和器件寄存器比特字段布局中硬件变化。

## Hardware Manager (HW Manager)

Hardware Manager (HW Manager)组件提供一组功能 API，解决了所选 HPS 资源的更复杂的配置和操作控制方面。

HW Manager 功能具有以下特点：

- 功能采用由 SoCAL 提供的低级别器件操作的组合，SoCAL 是以特定顺序执行的，以进行所需要的操作。
- 功能可以采用跨功能(例如来自不同的 IP 模块)的器件操作以实现要达到的效果。
- 对于预期器件响应的操作和验证的应用，功能可能需要满足指定的时序约束。
- 功能通过参数约束和验证检查来提供一定程度的用户保护和错误诊断。

HW Manager 功能是使用 SoCAL API 提供的单元操作实现的， 以实现更复杂的功能和服务。HW Manager 功能也可以通过 HW Manager API 中的其它功能的复合应用来实现，以构建更复杂的操作(例如 FPGA 的软件控制的配置)。

# 硬件库参考文档

SoCAL API 和 HW Manager API 的参考文档是作为 SoCEDS Toolkit 的一部分发布的。此参考文档作为可访问的在线 HTML 提供的，可以通过任何的网页浏览器进行访问。

在线 SoC FPGA Hardware Library (HWLIB)参考文档位于：

- SoC Abstraction Layer (SoCAL) API 参考文档：
  - Cyclone V and Arria V: <*SoC EDS installation directory*> **/ip/altera/hps/altera_hps/doc/soc_cv_av/socal/html/index.html**
  - Arria 10: <*SoC EDS installation directory*> **/ip/altera/hps/altera_hps/doc/soc_a10/socal/html/index.html**
- Hardware Manager (HW Manager) API 参考文档： <*SoC EDS installation directory*> **/ip/altera/hps/altera_hps/doc/hwmgr/html/index.html**

# 系统存储器映射

HPS hard IP 模块的地址可以通过所提供的 SoCAL 宏(macro)进行访问。SoCAL 也提供宏来访问 HPS hard IP 模块的单独寄存器和寄存器域。

对于 FPGA IP 模块，用于访问 IP 寄存器和寄存器域的宏通常是 IP deliverables 的一部分。然而，在 Qsys 工具中，基于实际 IP 模块的地址在系统集成时会经常改变。

工具"sopc-create-header-files"能用于创建一个 C include 文件，包括位于 FPGA 架构中的所 IP 模块的基地址。

**注意:** 此工具是 Quartus Prime 的一部分，而不是 SoC EDS 的一部分。安装 Quartus Prime 之后，就可以从 SoC EDS Embedded Command Shell 调用此工具。

此工具的基本用法是通过.sopcinfo 文件作为一个参数对其进行调用。例如，要生成 Arria 10 GHRD 的 include 文件，需要在此文件夹中执行下面命令：`sopc-create-header-files ghrd_10as066n2.sopcinfo`.

此工具对系统中的每个主器件(master)分别成一个 include 文件，从主器件的角度显示系统地址。因为文件**<hps_component_name>_a9_0.h** 从 HPS 角度显示系统地址，因此它可用于 HPS 软件开发。

下面示例展示了如何使用此工具生成一个显示 HPS A9 Core 0 角度的唯一文件：

```
sopc-create-header-files ghrd_10as066n2.sopcinfo --module\
arria10_hps_0_arm_a9_0 --single arria10_hps_0_arm_a9_0.h
```

此示例只创建一个文件，称作"arria10_hps_0_arm_a9_0.h"。

您也可以运行"`sopc-create-header-files --help`"获得关于此工具的更多信息，或者参考 Quartus Prime 文档。

# 硬件库文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | • 更新了 HW Library 图。<br>• 增添了新章节"系统存储器映射"。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# HPS Flash Programmer 用户指南

9

2016.02.17

ug-1137 订阅 反馈

Altera Quartus Prime 软件和 Quartus Prime Programmer 包括 HPS flash programmer。诸如 HPS 的硬件设计都在电路板上装有闪存，用于存储 FPGA 配置数据或 HPS 程序数据。HPS flash programmer 将数据编程到一个与 Altera SoC 连接的闪存器件中。programmer 通过 Altera 下载电缆(例如 USB-Blaster™ II)发送文件内容到 HPS，并指示 HPS 写入数据到闪存中。

HPS flash programmer 将以下类型的数据内容编程到闪存器件中：

- HPS 软件可执行文件—很多系统使用闪存来存储非易失性程序代码或固件。HPS 系统能够从闪存进行引导。

  **注意：** HPS Flash Programmer 主要目的是用于将 Preloader 映像编程到 QSPI 或 NAND flash 中。由于运行的速度很慢，不建议 HPS Flash Programmer 用于编程大尺寸文件。
- FPGA 配置数据—系统上电时，电路板上的 FPGA 配置控制器或者 HPS 读取闪存中的 FPGA 配置数据来编程 FPGA。配置控制器或者 HPS 可能能够在存储在闪存中的多个 FPGA 配置文件之间进行选择。
- 任何其他数据文件—根据不同的目的，HPS flash programmer 可将一个二进制文件编程到一个闪存中的任意位置。例如，一个 HPS 程序能够使用此数据作为一个系数表或中正弦查找表。

HPS flash programmer 编程以下类型的存储器：

- Quad serial peripheral interface (QSPI) Flash
- Open NAND Flash Interface (ONFI) compliant NAND Flash

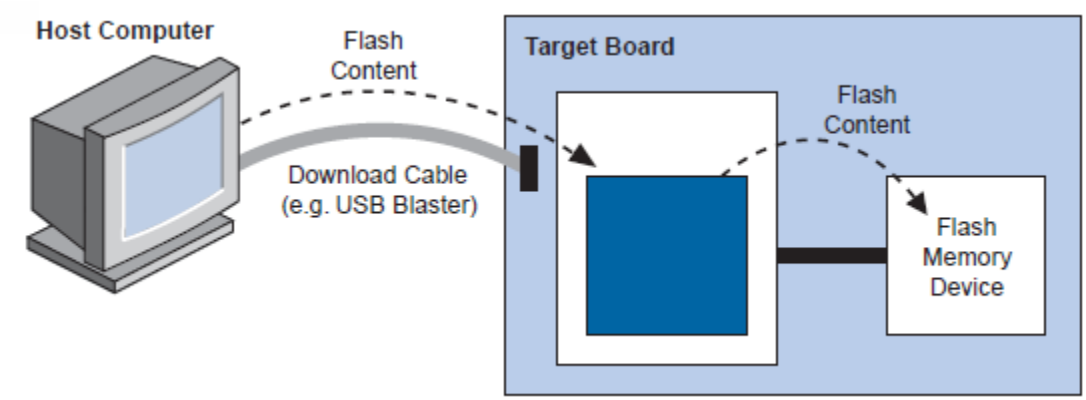
## HPS Flash Programmer 命令行工具

您可以直接从命令行运行 HPS flash programmer。对于 Quartus Prime 软件，HPS flash programmer 位于<*Altera installation directory*> **/quartus/bin**。对于 Quartus Prime Programmer，HPS flash programmer 位于<*Altera installation directory*> **/qprogrammer/bin**。

## HPS Flash Programmer 如何工作

HPS flash programmer 分成一个主机(host)和一个目标机器(target)。host 部分在您的计算机上运行，并通过下载电缆发送闪存编程文件和编程指令到 target。target 部分在 SoC 的 HPS 中。target 接收由 host 发送的编程数据闪存内容和关于目标闪存器件的所需信息。target 将数据写入到闪存器件中。

**图 9-1: HPS Flash Programmer**




HPS flash programmer 通过冷复位期间采集引导选择(BSEL)管脚来决定要编程的闪存类型；您不需要指定要编程的闪存类型。

# 从命令行使用 Flash Programmer

## HPS Flash Programmer

HPS flash programmer 工具能够对闪存进行擦除，空白检查，编程，验证和检验。此工具接受以"**.bin**"为扩展名的二进制文件。

HPS flash programmer 命令行语法：

```
quartus_hps <options> <file.bin>
```

**注意:** HPS flash programmer 使用字节寻址。

**表 9-1: HPS Flash Programmer 参数**

| 选项 | 短选项 | 是否需要 | 说明 |
|---|---|---|---|
| `--cable` | `-c` | Yes | 此选项指定要使用的下载电缆。<br>通过运行命令 "jtagconfig"来获得编程电缆的列表。此列表将列出可用的电缆，如下例所示：<br>jtagconfig<br>1) USB-Blaster [USB-0]<br>2) USB-Blaster [USB-1]<br>3) USB-Blaster [USB-2]<br>"-c"参数可以是编程电缆的编号或名称。下面是上述情况的有效示例：<br>-c 1<br>-c "USB-Blaster [USB-2]" |
| `--device` | `-d` | Yes (如果链中有多个 HPS 器件) | 此选项指定 HPS 器件的索引。此工具将自动检测链并决定 HPS 器件的位置；然而，如果链中有多个 HPS 器件，那么必须指定目标器件索引。 |
| `--operation` | `-o` | Yes | 此选项指定要执行的操作。下面的操作是被支持的：<br>• I：读取 SOC 器件的 IDCODE 并发现 Access Port<br>• S：读取闪存的 Silicon ID<br>• E：擦除闪存<br>• B：空白检查闪存<br>• P：编程闪存<br>• V：验证闪存<br>• EB：擦除并空白检查闪存<br>• BP：编程<*BlankCheck*>闪存<br>• PV：编程并验证闪存<br>• BPV: 编程(空白检查)并验证闪存<br>• X：检验闪存<br>**注意:** 程序在默认情况下编程闪存之前要先擦除闪存。 |
| `--addr` | `-a` | Yes (如果起始地址不是 0) | 此选项指定要执行的操作的起始地址。 |
| `--size` | `-s` | No | 此选项指定操作要执行的数据字节的数量。`size` 是可选的。 |

| 选项 | 短选项 | 是否需要 | 说明 |
|---|---|---|---|
| `--repeat` | `-t` | No | 这些选项必须一起使用。HPS BOOT 流程最多支持四个映像，其中每个映像都是相同的，这些选项复制操作数据；因此您不需要 eSW 来创建一个包含重复映像的大尺寸文件。<br>`repeat` 对要执行的操作指定重复映像的数量。<br>`interval` 指定重复的地址。默认值为 64 kilobytes (KB)。<br>`repeat` 和 `interval` 是可选的。 |
| `--interval` | `-i` | | |

## HPS Flash Programmer 命令行工具示例

输入 `quartus_hps --help` 可以获得关于如何使用的信息。您也可以输入 `quartus_hps --help=<option>`来获得关于每个选项的更多信息。例如"quartus_hps --help=o"。

### 实例 9-1: 将文件编程到闪存的地址 0 中

`quartus_hps -c 1 -o P input.bin` 使用电缆 *M* 将输入文件(**input.bin**)编程到闪存中，从闪存地址 0 开始。

### 实例 9-2: 将文件的头 500 个字节编程到闪存中(十进制)

`quartus_hps -c 1 -o PV -a 1024 -s 500 input.bin` 使用电缆 *M* 将输入文件(**input.bin**) 的头 500 个字节编程到闪存中，开始于闪存地址 1024，然后进行验证。

**注意:** 如果闪存地址没有前缀 0x，则假设为十进制。

### 实例 9-3: 将文件的头 500 个字节编程到闪存中(十六进制)

`quartus_hps -c 1 -o PV -a 0x400 -s 500 input.bin` 使用电缆 *M* 将输入文件(**input.bin**) 的头 500 个字节编程到闪存中，开始于闪存地址 1024，然后进行验证。

**注意:** 如果闪存地址有前缀 0x，则假设为十六进制。

### 实例 9-4: 将文件编程到闪存，在每 1 MB 重复两次

`quartus_hps -c 1 -o BPV -t 2 -i 0x100000 input.bin` 使用电缆 *M* 将输入文件(**input.bin**)编程到闪存中。在每 1 兆字节(MB)的闪存地址，此操作重复两次。在编程操作前，此工具确保闪存是空的。编程操作后，此工具验证已编程的数据。

### 实例 9-5: 擦除闪存地址上的闪存数据

`quartus_hps -c 1 -o EB input.bin` 使用电缆 *M* 擦除闪存地址上的闪存数据，输入文件(**input.bin**)存储在此闪存地址上，然后进行空白检查。

**实例 9-6: 擦除整个芯片**

`quartus_hps -c 1 -o E` 使用电缆 *M* 存储整个芯片。 如果没有指定输入文件(**input.bin**)时，则擦除所有闪存数据。

**实例 9-7: 擦除指定的闪存数据**

`quartus_hps -c 1 -o E -a 0x100000 -s 0x400000` 擦除指定的闪存数据。例如，使用电缆 *M* 擦除开始于 1 MB 的闪存地址上的 4 MB 存储器数据。

**实例 9-8: 检验闪存数据**

`quartus_hps -c 1 -o X -a 0x98679 -s 56789 output.bin` 使用电缆 *M* 从闪存地址 0x98679 开始检验 56789 字节的闪存数据。

# 支持的存储器件

**表 9-2: QSPI Flash**

| 闪存器件 | 模式 | 器件 ID | 芯片编号(DIE #) | 密度(Mb) |
|---|---|---|---|---|
| N25Q128A | Micron | 0x18BA20 | 1 | 128 |
| N25Q256 | Micron | 0x19BA20 | 1 | 256 |
| N25Q512 | Micron | 0x20BA20 | 2 | 512 |
| N25Q00 | Micron | 0x21BA20 | 4 | 1024 |
| N25Q128A | Micron | 0x18BB20 | 1 | 128 |
| N25Q256 | Micron | 0x19BB20 | 1 | 256 |
| N25Q512 | Micron | 0x20BB20 | 2 | 512 |
| lN25Q00 | Micron | 0x21BB20 | 4 | 1024 |
| | Micron | 0x132020 | 1 | |
| S25FL128S | Spansion | 0x182001 | 1 | 128<br>(64 KB 的扇区大小) |
| S25FL128S | Spansion | 0x182001 | 1 | 128<br>(256 KB 的扇区大小) |

| 闪存器件 | 模式 | 器件 ID | 芯片编号(DIE #) | 密度(Mb) |
|---|---|---|---|---|
| S25FL256S | Spansion | 0x190201 | 1 | 256<br>(64 KB 的扇区大小) |
| S25FL256S | Spansion | 0x190201 | 1 | 256<br>(256 KB 的扇区大小) |
| S25FL512S | Spansion | 0x200201 | 1 | 512 |

**表 9-3: ONFI 符合的 NAND Flash**

| 制造商 | MFC ID | 器件 ID | 密度(Gb) |
|---|---|---|---|
| Micron | 0x2C | 0x68 | 32 |
| Micron | 0x2C | 0x48 | 16 |
| Micron | 0x2C | 0xA1 | 8 |
| Micron | 0x2C | 0xF1 | 8 |

**注意:** HPS Flash Programmer 支持 HPS QSPI Flash Controller 支持的所有 ONFI 符合的 NAND 闪存器件。

**注意:** HPS Flash Programmer 支持 Cyclone V，Arria V 和 Arria 10 SoC 器件。

# HPS Flash Programmer 用户指南文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 在"支持的存储器件"章节中，在 QSPI Flash 表中添加了 QSPI Flash 部件编号。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# 裸机编译器(Bare Metal Compiler) 10

2016.02.17

ug-1137 订阅 反馈

随 SoC EDS 一起发布的裸机编译器为 Mentor Graphics Sourcery™ CodeBench Lite Edition, version 4.9.2。

关于 Sourcery CodeBench 精简版和最新版本下载的详细信息，请参考 Mentor Graphics 网站。

此编译器是一种基于 GCC 的 **arm-altera-eabi** 端口。它针对 ARM 处理器，假定裸机操作，并使用标准的 ARM 嵌入式应用程序二进制接口(EABI)约定。

裸机编译器作为 SoC EDS 安装进程的一部分被安装在以下文件夹中：<*SoC EDS 安装目录*> **/host_tools/mentor/gnu/arm/baremetal**。

嵌入式命令壳(通过从 *SoC EDS 安装目录*运行脚本打开)设置正确的环境 PATH 变量，以调用裸机编译工具。开始 shell 后，诸如 **arm-altera-eabi-gcc** 的命令能够被直接调用。当从嵌入式命令壳开始 ARM DS-5 AE 环境时，它继承环境设置，并能够直接调用这些编译工具。

或者，可以使用编译工具的完整路径： <*SoC EDS 安装目录*> **/host_tools/mentor/gnu/arm/baremetal/bin**.

裸机编译器带有完整文档，位于<*SoC EDS 安装目录*> **/host_tools/mentor/gnu/arm/baremetal/share/doc/sourceryg++-arm-altera-eabi**。此文档有四种不同的格式，以满足不同的用户需求。

- HTML 文件
- Info 文件
- Man 页面
- PDF 文件

所提供的文档有:

- 编译器手册
- 汇编手册
- Linker 手册
- Binutils 手册
- GDB 手册
- 入门指南(2)
- 库文件手册

**相关链接**

- **Mentor Graphics**

(2) 入门指南位于 Altera Wiki 上的 **SoCEDSGettingStarted** 页面。

- SoC EDS 入门指南

# 裸机编译器文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 更新了 Mentor Graphics Sourcery CodeBench Lite Edition 的编译器版本。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# SD 卡引导工具 11

2016.02.17

ug-1137 订阅 反馈

SoC EDS SD 卡引导工具是一种用于更新 SD 卡上的引导软件的工具。

Preloader 通常存储在 SD 卡上一个自定义分区中( type = 0xA2)。另外，下一个引导阶段(通常是 Bootloader)也可以存储在相同的自定义分区中。

因为它是一个自定义分区，所以如果没有文件系统，Preloader 和/或 Bootloader 不能通过复制新文件到卡上进行更新；需要一个软件工具。

SD 卡引导工具使用户能够更新物理 SD 卡或磁盘映像文件上的 Preloader 和/或 Bootloader。此工具不能用于创建一个新的可引导 SD 卡或磁盘映像文件。要想创建一个新的可引导 SD 卡或磁盘映像文件，建议您使用 Linux host OS 上的 fdisk。

**相关链接**

**Linux 软件开发工具** (第 12-1 页)

## 使用情况

此工具用于更新引导软件，此引导软件位于：

- 现有的 SD 卡
- 现有的磁盘映像文件

此管脚支持更新：

- Cyclone V 和 Arria V Preloader
- Cylone V 和 Arria V Bootloader
- Arria 10 Bootloader

**注意:** 对于 Cyclone V 和 Arria V，Preloader 文件需要有 boot ROM 要求的 mkpimage header，Bootloader 文件需要有 Preloader 要求的 mkpimage header。

**注意:** 对于 Arria 10，Bootloader 需要有 BootROM 要求的 mkpimage header。

**注意:** mkpimage 和 mkimage 工具都是 SoC EDS 的一部分。

此工具只更新存储在 Altera SoC 引导代码中的自定义分区，不会触及 SD 卡或磁盘映像文件的其他部分，包括 Master Boot Record (MBR)和任何其他分区(FAT, EXT3 等)以及可用空间。

**警告：**此工具的使用者需要有管理员或 root 访问权限才能使用此工具对物理 SD 卡进行写操作。如果只使用磁盘映像文件，那么就不需要这些权限。如果您没有这些相应的权限，请与您的 IT 部分取得联系。

# 工具选项

此工具是一个命令行程序。下表列出并描述了所有命令行选项；接下来的代码是此工具的`--help`输出。

**表 11-1: 命令行选项**

| 命令行参数 | 是否需要 | 说明 |
|---|---|---|
| **-p filename** | 可选的 | 指定要写入的 Preloader 文件 |
| **-b filename** | 可选的 | 指定要写入的 Cyclone V 或 Arria V Bootloader 文件 |
| **-B filename** | 可选的 | 指定要写入的 Arria 10 Bootloader 文件 |
| **-a write** | 需要 | 指定要进行的操作。仅支持"write"操作。例如："-a write" |
| **disk_file** | 需要(除非使用-d 选项) | 指定要写入的磁盘映像文件或物理磁盘。磁盘映像文件是一种包含存储卷的所有数据及分区表的文件。稍后可以使用其他工具将磁盘映像文件写入到物理磁盘中。对于 Linux 中的物理磁盘，只需指定器件文件。例如：对于 Windows 中的物理磁盘，/dev/mmcblk0 指定物理驱动器路径，例如：\\.\physicaldrive2，或者使用驱动器盘符选项(-d)指定一个驱动器盘符。在 Windows 中，驱动器盘符选项是最容易的方法。 |
| **-d** | 可选的 | 指定要写入的磁盘驱动器盘符。例如："-d E"。当使用此选项时，不能指定 disk_file 选项。 |
| **-h** | 可选的 | 显示帮助信息并退出 |
| **--version** | 可选的 | 显示脚本版本号并退出 |

### 此工具的输出样例

```
$ alt-boot-disk-util --help
Altera Boot Disk Utility
Copyright (C) 1991-2014 Altera Corporation

Usage:
#write preloader to disk
    alt-boot-disk-util -p preloader -a write disk_file

#write bootloader to disk
    alt-boot-disk-util -b bootloader -a write disk_file
```

```
#write BOOTloader and PREloader to disk
    alt-boot-disk-util -p preloader -b bootloader -a write disk_file

#write Arria10 Bootloader to disk
    alt-boot-disk-util -B a10bootloader -a write disk_file

#write BOOTloader and PREloader to disk drive 'E'
    alt-boot-disk-util -p preloader -b bootloader -a write -d E


Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -b FILE, --bootloader=FILE
                        bootloader image file'
  -p FILE, --preloader=FILE
                        preloader image file'
  -a ACTION, --action=ACTION
                        only supports 'write' action'
  -B FILE, --a10-bootloader=FILE
                        arria10 bootloader image file
  -d DRIVE, --drive=DRIVE
                        specify disk drive letter to write to
```

## SD 卡引导工具文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 维护版本。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# Linux 软件开发工具 12

2016.02.17

ug-1137 订阅 反馈

下面列出了 Linux 软件开发工具。Linux 编译器和 Linux 器件树生成器在本章接下来的部分有所描述。

- Linux 编译器
- SD 卡引导工具
- Linux 器件树生成器 (DTG)



**相关链接**

**SD 卡引导工具** (第 11-1 页)

## Linux 编译器

Linaro Linux 编译器(版本 4.8.3)随 SoC EDS 一起发布。

关于 Linux 编译器和最新版本下载的详细信息，请参考 Linaro 网站上的下载页面。

此编译器是一种基于 GCC 的 **arm-linux-gnueabihf** 端口。它针对 ARM 处理器，假定目标平台运行 Linux，并使用 GNU 嵌入式应用程序二进制接口(EABI)硬浮点(HF)约定。

Linux 编译器是作为 ARM DS-5 AE 的一部分进行安装的，而 ARM DS-5 AE 是作为 SoC EDS 的一部分进行安装的。l 编译工具位于<*SoC EDS installation directory*> **/ds-5/bin**。

Linux compiler 带有完整的文档，位于<*SoC EDS installation directory*> **/ds-5/documents/gcc**。这些文档以 HTML 文件提供，其中包括：

- 编译器手册
- 汇编手册
- Linker 手册
- Binutils 手册
- GDB 手册
- 入门指南

相关链接
Linaro Website

# Linux 器件树生成器(Linux Device Tree Generator)

器件树是一种数据结构，向操作系统(主要是 Linux 操作系统)描述底层硬件。通过将此数据结构传递到 OS 内核，一个单一的 OS 二进制可以支持硬件的多种类型。当硬件包括 FPGA 时，这种灵活性就显得格外重要。

Linux Device Tree Generator (DTG)工具是 Altera SoC EDS 的一部分，用于创建 SoC 系统的 Linux 器件树，SoC 系统包含使用 Qsys 创建的 FPGA 设计。生成的器件树描述了 HPS 外设，所选的 FPGA Soft IP 和那些依赖于电路板的 l 外设。

**注意:** Arria 10 Bootloader 也有一个相关联的 Device Tree，称作 Bootloader Device Tree，由 BSP Editor 工具创建并管理。

相关链接

- **Altera Wiki**
  关于 Linux DTG 的详细信息，请参考 Altera Wiki 网页。
- **Linux 器件树生成器用户指南**
  关于详细信息，请参考 Rocketboards 网站上 **Device Tree Generator Documentation** 页面上的 **Device Tree Generator User Guide**。

# Linux 软件开发工具文档修订历史

| 日期 | 版本 | 修顶内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 维护版本。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |

# 支持和反馈 13

2016.02.17

ug-1137 订阅 反馈

Altera 非常重视您的反馈意见。如果您想要报告软件漏洞，潜在的增强改进或者想要获得任何其他信息，那么请与您的 Altera TSFAE 取得联系或者提交一个服务请求。

**相关链接**

**myAltera Account Sign In**

## 支持和反馈文档修订历史

| 日期 | 版本 | 修订内容 |
|---|---|---|
| 2016 年 2 月 | 2016.02.17 | 维护版本。 |
| 2015 年 8 月 | 2015.08.06 | 增添了 Arria 10 支持。 |