ARM® Workbench IDE

4.0 版

用户指南

ARM DUI 0330EC

ARM Workbench IDE
用户指南

版本信息

本手册进行了以下更改。

更改历史记录

日期	发行号	保密性	变更
2006 年 3 月	A	非保密	ARM® RealView® Development Suite v3.0 版
2007 年 3 月	B	非保密	RealView Development Suite v3.1 版
2007 年 7 月	C	非保密	RealView Development Suite v3.1 版 - ARM Flash 编程器和汇编器编辑器更新
2007 年 12 月	D	非保密	RealView Development Suite v3.1 版 - CodeWarrior 导入程序更新
2008 年 9 月	E	非保密	RealView Development Suite v4.0 版

所有权声明

本文档描述的产品还将不断发展和完善。 ARM 将如实提供本文档所述产品的所有特性及其使用方法。 但是，所有暗示或明示的担保，包括但不限于对特定用途适销性或适用性的担保，均不包括在内。

本文档的目的仅在于帮助读者使用产品。 对于因使用本文档中的任何信息、文档信息出现任何错误或遗漏或者错误使用产品造成的任何损失或损害，ARM 公司概不负责。

使用 ARM 一词时，它表示“ARM 或其任何相应的子公司”。

保密状态

本文档的内容是非保密的。 根据 ARM 与 ARM 将本文档交予的参与方的协议条款，使用、复制和公开本文档内容的权利可能会受到许可限制的制约。

受限访问是一种 ARM 内部分类。

产品状态

本文档的信息是开发的产品的最新信息。

网址

http://www.arm.com

目录
ARM Workbench IDE
用户指南

前言

本前言简要介绍《*ARM Workbench IDE* 用户指南》。 本章分为以下几节：

关于本手册

本手册简要介绍 ARM® Workbench IDE，并介绍如何将其与 ARM RealView® Development Suite 中的其他工具一起使用。 Workbench 基于 Eclipse，但它专门用于为 ARM 目标生成、调试、监视和管理项目。

本手册的目的不是让用户熟悉 Workbench 的所有方面。 有关本指南中没有介绍的其他功能的信息，请参阅动态帮助中的标准《Workbench 用户指南》(*Workbench User Guide*) 或《C/C++ 开发用户指南》(*C/C++ Development User Guide*)。 有关详细信息，请参阅第2-24 页的*动态帮助*。

适用对象

本手册是为在 Microsoft Windows 或 Unix 下使用 Workbench 管理面向 ARM 的开发项目的所有开发人员编写的。 它假定您是一位有经验的软件开发人员，熟悉 RealView 工具。 它并不要求您熟悉 Workbench。

使用本手册

本手册由以下章节组成：

第 1 章 ***简介***

本章简要介绍 Workbench 及其附带的 ARM 插件。

第 2 章 ***使用入门***

本章提供有关生成过程、生成配置、Workbench 功能、如何启动 Workbench 以及如何使用工作区的信息。

第 3 章 ***使用项目***

本章提供有关使用各种 ARM 项目类型，如何创建新项目、导入现有项目及向项目添加文件的信息。

第 4 章 ***配置生成和编译工具***

本章提供有关如何配置编译工具来控制 Workbench 生成项目方式的信息。

第 5 章 ***使用编辑器***

本章提供有关 Workbench 附带的各种编辑器的信息。

第 6 章 ***使用 ARM Flash 编程器***

本章提供有关如何在 Workbench 中配置和使用 Flash 设备的信息。

第 7 章 *使用 RealView Debugger*

本章提供有关如何在 Workbench 中连接和使用 RealView Debugger 的信息。

附录 A *术语、快捷键和图标*

本附录提供有关 Workbench 概念和含义的信息。

本手册假定 ARM 软件安装在缺省位置。 例如，在 Windows 上，安装位置可能是 *volume*:\Program Files\ARM。 在引用路径名时，这将作为 *install_directory* 的位置。 例如：

install_directory\RVDS\Examples\...

如果将 ARM 软件安装在其他位置，则需要更改此位置。

印刷约定

本手册使用以下印刷约定：

斜体	突出显示重要注释，介绍特殊术语，表示内部交叉参考和引用。
粗体	突出显示界面组件，如菜单名称。 表示 ARM 处理器信号名称。 必要时还用于说明列表中的术语。
`monospace`	表示可以从键盘输入的文本，如命令、文件和程序名以及源代码。
`mono`space	表示允许的命令或选项缩写。 可只输入下划线标记的文本，无需输入命令或选项的全名。
`monospace italic`	表示此处的命令和函数的变量可用特定值代替。
`等宽粗体`	表示在示例代码以外使用的语言关键字。

更多参考出版物

本部分列出了 ARM 公司和第三方发布的、可提供有关 ARM 系列处理器开发代码的附加信息的出版物。

ARM 将定期对其文档进行更新和更正。 有关最新勘误表、附录和 ARM *常见问题* (FAQ)，请访问 `http://infocenter.arm.com/help/index.jsp`。

ARM 出版物

有关 RealView Development Suite 各组件的详细文档，请参阅下列出版物：

- 《RealView Development Suite 入门指南》(ARM DUI 0255)
- 《RealView 编译工具要点指南》(ARM DUI 0202)
- 《RealView 编译工具开发指南》(*RealView Compilation Tools Developer Guide*) (ARM DUI 0203)
- 《RealView 编译工具汇编程序指南》(*RealView Compilation Tools Assembler Guide*) (ARM DUI 0204)
- 《RealView 编译工具编译器用户指南》(*RealView Compilation Tools Compiler User Guide*) (ARM DUI 0205)
- 《RealView 编译工具编译器参考指南》(ARM DUI 0348)
- 《RealView 编译工具链接器用户指南》(ARM DUI 0206)
- 《RealView 编译工具链接器参考指南》(ARM DUI 0381)
- 《RealView 编译工具实用程序指南》(ARM DUI 0382)
- 《RealView 编译工具库和浮点支持指南》(ARM DUI 0349)
- 《RealView Debugger 要点指南》(*RealView Debugger Essentials Guide*) (ARM DUI 0181)
- 《RealView Debugger 用户指南》(*RealView Debugger User Guide*) (ARM DUI 0153)。

其他出版物

本手册介绍 ARM 提供的特定于 Workbench 的信息。 有关 Eclipse 的详细信息，请访问 Eclipse 网站，地址为 `http://www.eclipse.org`。

反馈

ARM Limited 欢迎用户就 Workbench 及其文档提供反馈。

有关 Workbench 的反馈

如果对 Workbench 有任何问题，请与供应商联系。 为便于供应商快速提供有用的答复，请提供以下信息：

- 您的姓名和公司名称。
- 产品的序列号和版本号。
- C/C++ 开发工具和 Workbench 的版本号。 若要获取这些信息，请选择 **Help/帮助 → About ARM Workbench IDE/关于 ARM Workbench IDE**，然后单击 **Plug-in Details/插件详细信息**。
- 所有已安装组件的详细信息和版本号，如硬件平台、操作系统、GNU make 和 JRE。
- 能重现问题的一小段独立的代码。
- 您期望发生和实际已经发生的情况的详细说明。
- 您使用的命令，包括所有命令行选项。
- 能说明问题的示例输出。
- 工具的版本字符串，包括版本号和内部版本号。

关于本手册的反馈

如果您对本手册有任何疑问，请发送电子邮件至 errata@arm.com，并提供：

- 文档标题
- 文档号
- 您有疑问的页码
- 问题的简要说明

我们还欢迎您对需要增加和改进之处提出建议。

第 1 章
简介

本章概述 Workbench 及其主要功能和安装要求。 本章分为以下几节：

- 第1-2 页的*关于 Workbench*
- 第1-3 页的*关于 ARM 插件*
- 第1-4 页的*安装要求*

有关具体概念、含义和用法的详细信息，请参阅附录 A *术语、快捷键和图标*。

1.1 关于 Workbench

Workbench 是一种*集成开发环境* (IDE)，它将软件开发与 ARM® RealView® 工具的编译和调试技术结合在一起。 它可以用作项目管理器，为 ARM 目标创建、生成、调试、监视和管理项目。 它使用一个称为“工作区”的文件夹来存储与特定项目相关的文件和文件夹。

有关详细信息，请参阅第 2 章 *使用入门*。

1.2 关于 ARM 插件

Workbench 集成了下列 ARM 插件：

RealView 编译工具

通过此插件，可以在 Workbench 中使用 RealView 编译工具为 ARM 目标生成项目。 它提供了综合配置面板，用于修改项目和各文件的工具设置。

ARM 汇编器编辑器

此插件提供了一个编辑器，以便于阅读的可自定义代码格式显示 ARM 编译器文件。 它还为标签及其他导航辅助工具提供自动完成功能。

属性编辑器

此插件为 ARM 汇编器和 C/C++ 编辑器提供扩展。 您可以配置源代码来提供 GUI 组件，这样无需直接编辑代码即可修改变量或 `#defines`。

分散文件编辑器

此插件提供了一个编辑器，使您可以轻松地创建和编辑分散加载描述文件。

ELF 内容编辑器

此插件创建表格式窗体和图形视图，用于显示映像文件、对象文件和库文件的内容。

ARM Flash 编程器

此插件提供了一个新的项目向导，用于为目标创建 Flash 算法和程序映像。 它还提供相关的导出向导，实现与 RealView Debugger 的紧密集成。

有关上述每个插件的详细信息，请参阅：

- 第 4 章 *配置生成和编译工具*
- 第 5 章 *使用编辑器*
- 第 6 章 *使用 ARM Flash 编程器*

此外，您还可以使用动态帮助：

1. 从 **Help/帮助**菜单中选择 **Help Contents/帮助目录**。
2. 在“Contents/目录”框架中，选择 **ARM Workbench IDE Dynamic Help/ARM Workbench IDE 动态帮助**，然后选择要获得帮助的插件。

1.3 安装要求

需要安装下列组件才能使用 Workbench。 如果从 CD 安装 RealView Development Suite，则会安装所有必需的组件，包括 ARM Workbench IDE。

如果 Eclipse IDE 是自定义安装的，则在使用 ARM Workbench IDE 之前，必须确保已按如下顺序安装了下列组件。

Java 运行时环境 (JRE)

下载并安装 J2SE 5.0 或更高版本。 请参阅 `http://www.java.com` 或 `http://www.eclipse.org`。 必须使用 32 位版本的 JRE，即使在 64 位操作系统上运行也是如此。

Eclipse 从 `http://www.eclipse.org` 下载并安装 Eclipse v3.3。

BIRT 从 `http://www.arm.com/eclipse` 下载下列 BIRT 功能：

- 图表和报告创建
- 数据库开发

CDT *C 和 C++ 开发工具* (CDT) 是一种插件，可将 C 和 C++ 生成环境集成到 IDE 中。 您可以将此插件作为 Eclipse+CDT 捆绑产品的一部分进行安装，网址是 `http://www.eclipse.org`。 如果要使用最新的 RealView 工具，则需要安装 CDT v4.0.3。

ARM 插件

使用 Workbench 的软件更新功能来安装 ARM 插件，网址是 `http://www.arm.com/eclipse`。 请参阅第2-26 页的*安装新功能*。

GNU 工具链

如果要与 GNU 工具链集成，则必须安装下列工具之一：

- MinGW，网址是 `http://www.mingw.org`
- Cygwin，网址是 `http://www.cygwin.com`

第 2 章
使用入门

本章介绍 Workbench、C/C++ 透视图和相关功能。 还介绍如何配置 Workbench、如何生成项目，以及如何使用 Workbench 所提供的不同类型的帮助。

本章分为以下几节：

2.1 启动 Workbench

第一次启动 Workbench 时，请接受缺省工作区，单击 **OK/确定**打开初始“Welcome/欢迎”视图，如图 2-1 所示。 如果愿意，您可以在以后选择其他工作区位置，有关详细信息，请参阅第2-5 页的*工作区*。

现在，您可以访问以下主题：

概览 Workbench 和相关工具的简介。

新增功能 新增功能更新信息的网站链接。

教程 特定主题的分步指南。

Workbench 用于创建、管理和生成项目的开发环境。

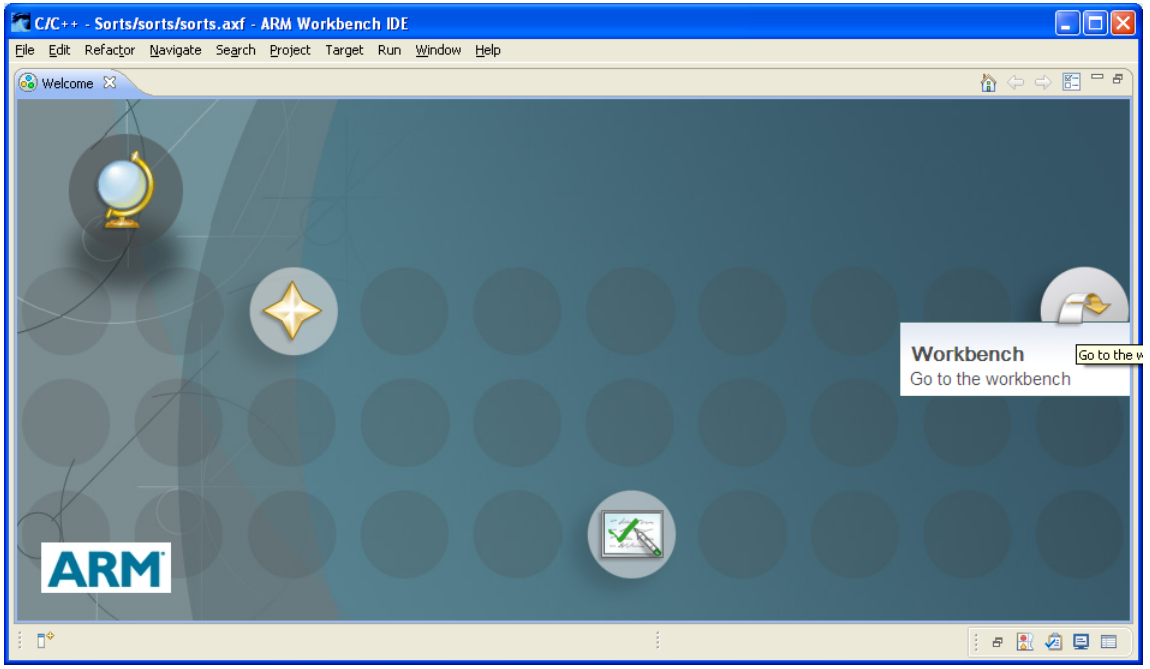

图 2-1 “Welcome/欢迎”视图

2.1.1 语言包

Workbench 的某些功能已翻译为不同语言。 ARM Workbench IDE 只为 ARM 插件提供语言翻译。 您可以从 Eclipse 网站下载其他可用的语言包。

可通过两种方式使用另一种语言包启动 Workbench：

- 如果操作系统的语言正是您需要使用的语言，则 Workbench 自动以翻译语言显示功能。
- 如果操作系统的语言不是您需要使用的语言，则必须在启动 Workbench 时指定 `-nl` 命令行参数。

  例如，若要使用日语语言包，可以使用：

  ```
  awide-4.0 -nl ja
  ```

ARM Workbench IDE 提供以下语言翻译：

ja 日语
ko 朝鲜语
zh_CN 简体中文

2.1.2 打开和关闭 Workbench 窗口

若要打开 Workbench 窗口，请单击标记为 **Workbench** 的弯曲箭头图标，请参阅第2-2 页的图 2-1。

注意

通过从 **Help/帮助**菜单中选择 **Welcome/欢迎**，随时可以返回“Welcome/欢迎”视图。

第2-4 页的图 2-2 是一个典型的 Workbench 窗口，其中显示的是 C/C++ 透视图和一些关联视图。

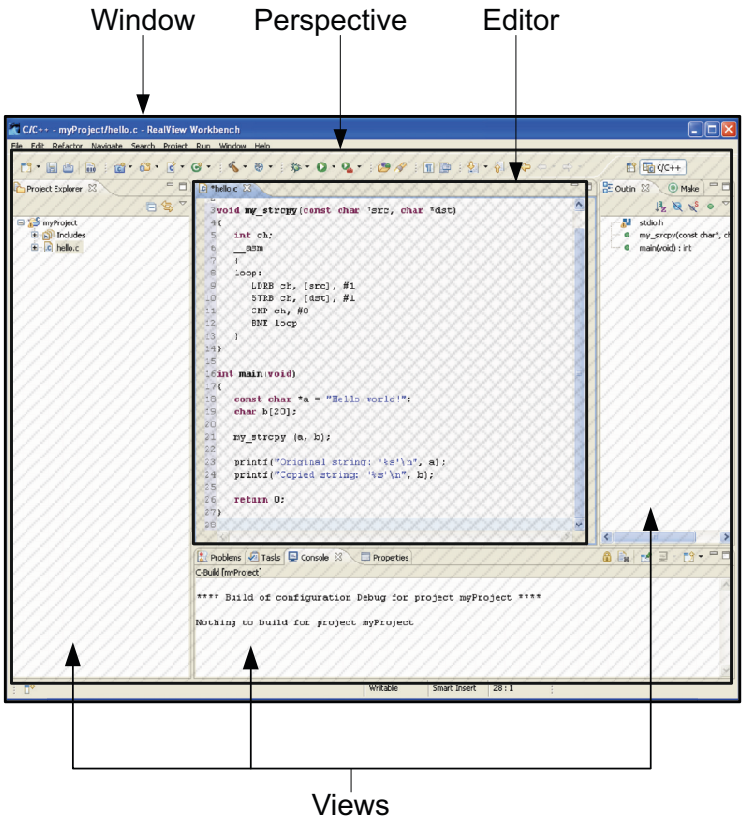

图 2-2 Workbench 窗口

若要关闭 Workbench 窗口并退出，请从 **File/文件**菜单中选择 **Exit/退出**，或单击窗口顶角的关闭图标。 退出时，Workbench 会自动保存，以便在下次打开 Workbench 时，通过使用保存的设置，窗口依然是同样的透视图和视图。

2.2 Workbench 功能

Workbench 窗口是主开发环境，用于管理各个项目、关联的子文件夹和源文件。 每个 Workbench 窗口都链接到一个工作区。 如果要同时使用不同的工作区，可以启动多个 Workbench 窗口并将每个窗口都链接到不同的工作区。 有关详细信息，请参阅*工作区*。

本节介绍主要的 Workbench 功能：

编辑器 编辑器是一些特殊类型的视图，用于显示相关源文件。 编辑器区域中的选项卡显示当前打开进行编辑的文件。 有关详细信息，请参阅第 5 章 *使用编辑器*。

菜单和工具栏

主菜单和工具栏位于 Workbench 窗口的顶部。 与特定功能关联的其他工具栏位于每个透视图或视图的顶部。 有关详细信息，请参阅第2-12 页的*菜单*和第2-13 页的*工具栏*。

透视图 透视图定义 Workbench 中所选视图和编辑器的布局。 透视图也有自己的关联菜单和工具栏。 有关详细信息，请参阅第2-10 页的*透视图和视图*。

资源 资源是 Workbench 中的项目、文件和文件夹。 有关详细信息，请参阅第2-7 页的*资源*。

视图 视图提供与编辑器中的活动文件关联的相关信息。 视图也有自己的关联菜单和工具栏。 有关详细信息，请参阅第2-10 页的*透视图和视图*。

工作区 工作区是在文件系统上指定的区域，用于存储与 Workbench 项目和个人 Workbench 设置有关的文件和文件夹。 有关详细信息，请参阅*工作区*。

2.2.1 工作区

工作区是在文件系统上指定的区域，用于存储与 Workbench 项目和个人 Workbench 设置有关的文件和文件夹。

——注意——

建议为 Workbench 项目选择专用工作区文件夹。 如果选择包含非相关资源的现有文件夹，则无法在 Workbench 中访问这些资源。 以后创建和生成项目时，这些资源还可能导致冲突。

在“Preferences/首选项”对话框中对自定义设置所做的更改保存在工作区中。如果选择另一个工作区，这些设置可能不同。

第一次启动 Workbench 时，将打开用于选择工作区的“Workspace Launcher/工作区启动器”对话框，请参阅图 2-3。

以后启动时，上次保存的工作区会在下拉框中显示为缺省选择。 通过单击向下箭头或 **Browse.../浏览...** 按钮，可以选择其他工作区。

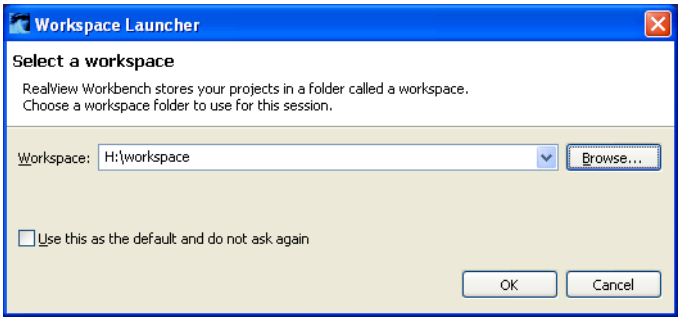

图 2-3 “Workspace launcher/工作区启动器”对话框

如有必要，可选中复选框，以便在以后启动时禁止打开“Workspace Launcher/工作区启动器”对话框。

注意

通过从 **File/文件**菜单中选择 **Switch Workspace.../切换工作区...**，随时可以更改缺省工作区。

若要打开 Workbench 并自动链接到特定工作区，可以使用 `-data` 命令行参数。例如：

```
awide-4.0 -data h:\workspace
```

在 Workbench 外部编辑文件

即使没有运行 Workbench，也可以编辑项目子文件夹和文件。 下次启动 Workbench 时，缺省首选项会使相关视图进行刷新和更新。 如果更改缺省首选项，可以在“Project Explorer/项目资源管理器”视图中单击更新后的子文件夹或文件，然后从“File/文件”菜单中选择 **Refresh/刷新**。

2.2.2 资源

资源是一个通用术语，用于描述项目、文件、文件夹或这些内容的组合。 资源在 Workbench 之中，但可能不会始终在工作区中。 资源的类型有三种：

项目 项目显示在“Project Explorer/项目资源管理器”视图中，可以存储在工作区文件夹中，也可以是链接资源。 有关详细信息，请参阅*链接资源*。

项目必须在 Workbench 中，才能将其他资源导入或链接到该项目。 在项目创建过程中，会创建附加配置文件和文件夹（例如生成属性）。 不能编辑或删除这些附加文件和文件夹。

文件夹 文件夹显示在“Project Explorer/项目资源管理器”视图中，可以位于工作区文件夹中，也可以是链接资源。 有关详细信息，请参阅*链接资源*。

文件 文件显示在“Project Explorer/项目资源管理器”视图中，可以位于工作区文件夹中，也可以是链接资源。 有关详细信息，请参阅*链接资源*。

链接资源

资源可以在项目之间共享，也可以存在于所选工作区之外的文件夹系统中。 为此，必须在 Workbench 中创建链接。

——注意——

链接文件和文件夹必须将一个项目作为其父资源。

删除、移动或复制链接资源只会影响 Workbench 中的链接，而不会影响链接到的资源。 但是，从链接文件夹中删除子资源也会从文件夹系统中将其删除！

链接文件

若要将现有文件链接到工作区中的项目而不是复制该文件，可以使用“New File/新建文件”向导的高级设置。 缺省情况下，第2-8 页的图 2-4 中所示的高级选项不可见，单击 **<<Advanced/高级**按钮可显示这些选项。 路径变量也可用来引用文件。 有关使用变量的详细信息，请参阅动态帮助。

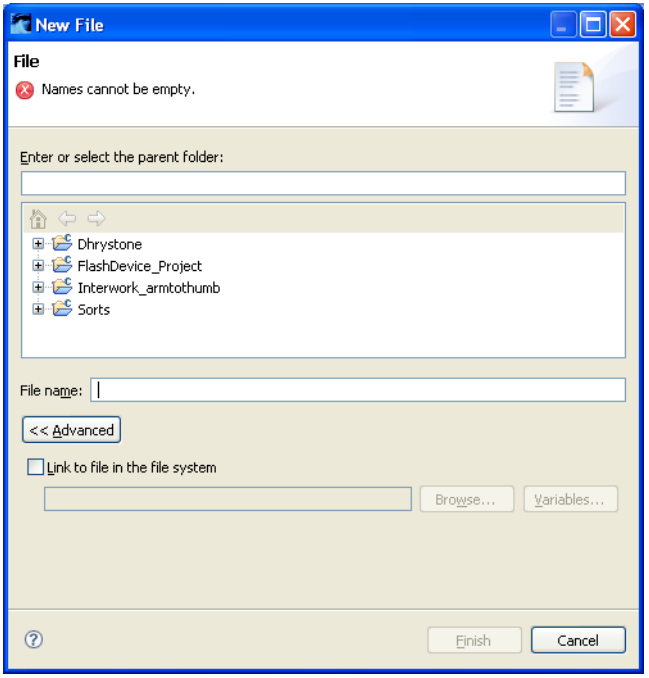

图 2-4 链接文件

有关创建新文件的详细信息，请参阅第3-15 页的*向项目中添加文件*。

链接文件夹

若要将现有文件夹链接到工作区中的项目而不是复制该文件夹，可以使用“New Folder/新建文件夹”向导的高级设置。 缺省情况下，第2-9 页的图 2-5 中所示的高级选项不可见，单击 **<<Advanced/高级**按钮可显示这些选项。 路径变量也可用来引用文件，有关详细信息，请参阅动态帮助。

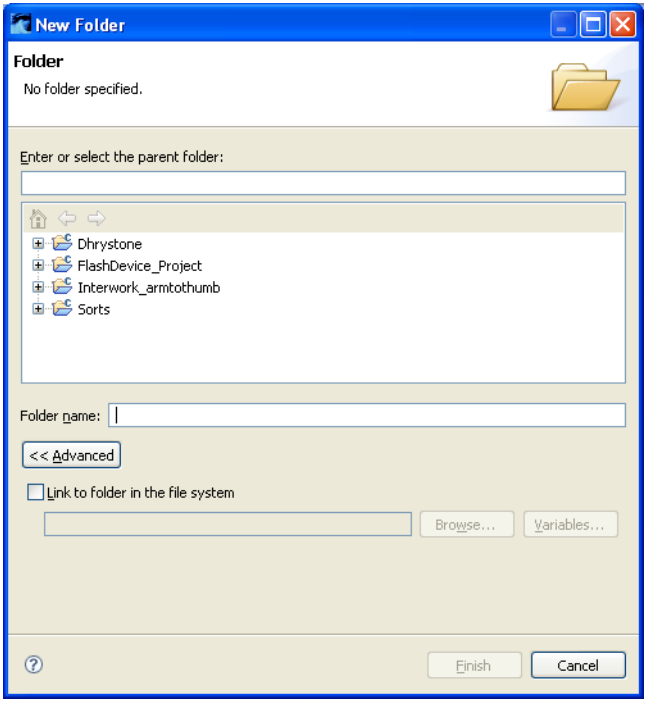

图 2-5 链接文件夹

链接项目

Workbench 使用“Import/导入”向导创建现有项目的链接。 如果有包含共享项目的中心文件夹，这会十分有用。 缺省情况下，第2-10 页的图 2-6 中所示的 **Copy projects into workspace/将项目复制到工作区中**复选框为选中状态，以确保将项目的副本放置到当前工作区中。 如果要使项目保留在原处并创建其链接，则必须取消选择此选项。

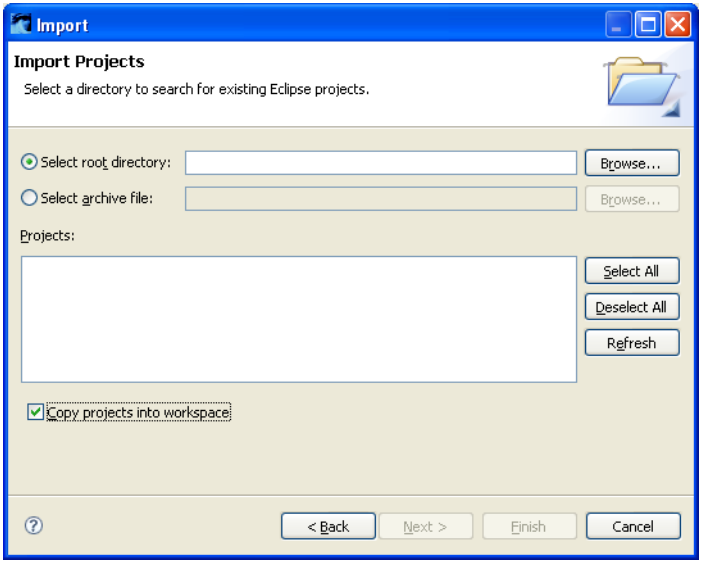

图 2-6 链接项目

有关导入现有项目的详细信息，请参阅第 3 章 *使用项目*。

禁用链接资源

通过在“Preferences/首选项”对话框中更改 **General/常规 → Workspace/工作区** 设置，可以禁用链接资源。 有关详细信息，请参阅第2-16 页的 *“Preferences/首选项”对话框*。

2.2.3 透视图和视图

主 Workbench 窗口包含一个或多个透视图，每个透视图都包含一个或多个视图。

透视图

对于所有 ARM 项目，都可以使用 C/C++ 透视图。

对于其他项目，通过使用第2-11 页的图 2-7 中所示的透视图工具栏，或从 **Window/窗口**菜单中选择 **Open perspective/打开透视图**，可以更改透视图。

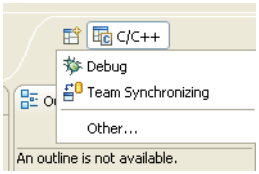

图 2-7 更改透视图

视图

视图是 Workbench 中的小型可视化组件，用于在资源中导航或显示生成属性或编辑属性。 编辑器是特殊类型的视图，用于查看和编辑源文件。 以下视图与 Workbench 相关联：

Project Explorer/项目资源管理器

此视图类似于“Navigator/导航器”视图和“C/C++ Projects/C/C++ 项目”视图，它提供 C/C++ 项目文件的相关资源的层次结构视图。 如果右击某个资源，则会出现特定任务的上下文菜单。

Console/控制台

此视图为程序提供 I/O 接口，以实现键盘输入和文本输出。

Editor/编辑器

编辑器与特定文件类型相关联，在“Project Explorer/项目资源管理器”视图中打开可编辑文件时，会自动打开相关编辑器视图。

Workbench 提供以下编辑器：

- 标准 C/C++ 编辑器
- ARM 汇编器编辑器
- 属性编辑器
- 分散文件编辑器
- ELF 内容编辑器

有关详细信息，请参阅第 5 章 *使用编辑器*。

Help/帮助 在选择问号图标时，此视图显示所选功能的动态帮助。 有关详细信息，请参阅第2-24 页的*动态帮助*。

Outline/大纲

此视图显示活动文件中 C/C++ 元素的结构化列表。 单击某个元素可将编辑器焦点更改为该元素在活动文件中的位置。

Problems/问题

此视图显示在生成过程中出现的错误消息。 选择错误消息会打开关联文件并将焦点移至导致问题的行。

Properties/属性

此视图显示所选项的名称和值。 例如，文件的最近修改时间/日期。

有关此处未列出的其他视图的详细信息，请参阅动态帮助。

2.2.4 菜单

主菜单位于 Workbench 窗口的顶部，可以按照个人喜好进行自定义。 其中的选项可能因所安装的插件和活动透视图而异。

Workbench 主菜单支持以下选项：

File/文件 用于创建、保存、关闭、打印、导入和导出资源。 还可以管理项目和文件属性设置。

Edit/编辑 用于剪切、复制、粘贴、查找和替换资源中的文本。

Refactor/重构

用于重命名代码中的所选对象，并在项目的其他文件中传播更改。

Navigate/导航

用于浏览和快速查找特定资源。

Search/搜索 提供高级筛选器来搜索资源。

Project/项目 用于管理项目生成配置并执行特定生成。 您还可以自定义 RealView 编译工具的工具设置。

Target/目标 用于管理 Flash 设备和 Flash 目标。

Run/运行 用于运行、发送至、调试或配置外部工具。 您还可以管理断点和观察点。

Window/窗口

用于打开、关闭和自定义透视图、视图和编辑器。

Help/帮助 提供 Workbench 和 RealView Development Suite 附带的工具的文档。 您还可以访问 ARM 备忘单和软件更新。

如果右击某个资源，则会出现特定任务的上下文菜单。 有关此处未列出的其他菜单选项的详细信息，请参阅动态帮助。

2.2.5 工具栏

主工具栏位于 Workbench 窗口的顶部，可以按照个人喜好进行自定义。图 2-8 是 C/C++ 透视图的主 Workbench 工具栏。 其中的选项可能因所安装的插件和活动透视图而异。

图 2-8 Workbench 工具栏

与特定功能关联的其他工具栏位于每个透视图或视图的顶部，请参阅图 2-9 和图 2-10。

图 2-9 透视图工具栏

图 2-10 视图工具栏

有关详细信息，请参阅第A-5 页的*菜单和工具栏图标*。

2.3 编辑源代码

您可以使用 Workbench 附带的编辑器来编辑源代码，也可以使用外部编辑器。如果使用外部编辑器，则必须刷新 Workbench 才能使视图与最近更新同步。为此，可以在“Project Explorer/项目资源管理器”视图中单击更新过的项目、子文件夹或文件，然后从 **File/文件**菜单中选择 **Refresh/刷新**。也可以通过在“Preferences/首选项”对话框中选择 **General/常规 → Workspace/工作区 → Refresh automatically/自动刷新**来启用自动刷新选项。有关详细信息，请参阅第2-16 页的*“Preferences/首选项”对话框*。

在 Workbench 中打开文件时，会出现一个新的编辑器选项卡，选项卡上有该文件的名称。如果文件经过了编辑，则选项卡名称中会显示一个星号 (*)，表示还未保存对该文件所做的更改。

打开两个或更多编辑器选项卡时，通过单击某个选项卡并将其拖到编辑器边框之上，可以并排显示选项卡。

在每个编辑器选项卡的左边距中，可以找到一个竖栏，其中显示与活动文件相关的标记。有关详细信息，请参阅第A-7 页的*编辑器标记*。

2.3.1 导航

在 Workbench 中，可通过多种方式浏览至特定资源。通过在资源树中浏览并双击某个文件，可以使用“Project Explorer/项目资源管理器”视图打开资源。另一种方式是使用快捷键或使用“Navigate/导航”菜单中的选项。

在“Navigate/导航”菜单中，可以通过“Go To/转到”选项按模式匹配查找资源，您也可以使用“Open Resource/打开资源”选项直接在编辑器中打开文件。

2.3.2 搜索

若要在 Workbench 中查找包含一个或多个文件中的信息或特定代码，可以使用“Search/搜索”菜单中的选项。在可自定义的“Search/搜索”对话框中，提供了按模式匹配的文本搜索和用于限定搜索字段的筛选器。您也可以从主 Workbench 工具栏中打开此对话框。

2.3.3 内容帮助

C/C++ 和 ARM 汇编器编辑器为光标位置处的函数和标签提供内容帮助，用于自动完成所选项。 使用快捷键 Ctrl+空格键时，会出现一个小对话框，其中包含可供选择的有效选项的列表。 在使用快捷键之前，通过键入部分字符可以缩短列表。 在列表中使用箭头键选择所需项，然后按 Enter 键即可将该项插入到代码中。

2.3.4 书签

书签可以用于标记文件中的特定位置或标记整个文件，以便您可以快速返回该位置或该文件。 若要创建书签，请选择要标记的文件或代码行，然后从 **Edit/编辑**菜单中选择 **Add Bookmark/添加书签**。 “Bookmarks/书签”视图显示用户定义的所有书签，在“Window/窗口”菜单中选择 **Show View/显示视图 → Bookmarks/书签**可以访问该视图。 如果未列出“Bookmarks/书签”视图，请选择 **Others.../其他...** 打开扩展列表。

若要删除书签，请打开“Bookmarks/书签”视图，单击要删除的书签，然后从“Edit/编辑”菜单中选择 **Delete/删除**。

2.4 配置 Workbench

通过更改布局、键绑定、文件关联和颜色方案可以将 Workbench 自定义为自己的设置。 有关详细信息，请参阅 *“Preferences/首选项”对话框*。

此外，还可以通过修改所选资源的属性来配置项目和文件，从而以不同的方式使用生成系统。 有关详细信息，请参阅第2-17 页的*属性对话框*。

使用“Window/窗口”菜单中的选项可以打开和自定义透视图，此外，还可以使用透视图菜单和工具栏。 缺省情况下，透视图会在同一窗口中打开，但是如果愿意，可以更改缺省首选项在新窗口中打开。

通过将视图拖放到所需位置，可以根据需要移动或停靠视图。 双击视图的标题栏可切换最大化/最小化选项，使用视图菜单和工具栏也可以实现这一功能。

2.4.1 “Preferences/首选项”对话框

使用“Preferences/首选项”对话框可以自定义 Workbench 设置，请参阅第2-17 页的图 2-11。 可以通过从 **Window/窗口**菜单中选择 **Preferences.../首选项...** 来访问此对话框。 对这些设置所做的更改会保存在当前工作区中。 如果要将 Workbench 设置复制到另一个工作区，请使用“Export/导出”向导，请参阅第2-21 页的*导入和导出*。

首选项层次结构树内容如下：

General/常规 控制工作区、生成顺序、链接资源、文件关联、路径变量、后台操作、键盘和鼠标设置。

ARM 控制 ARM 特定编辑器的外观和格式以及缺省设备属性。

C/C++ 控制 C/C++ 环境设置、CDT 生成变量、语法格式和缺省项目向导设置。

Help/帮助 控制上下文帮助的显示方式。

Install/Update/安装/更新

控制更新历史记录、计划程序和策略。

Run/Debug/运行/调试

控制在进行运行和调试之前的缺省透视图、断点、生成和启动设置。

Team/团队 控制 CVS 同步。

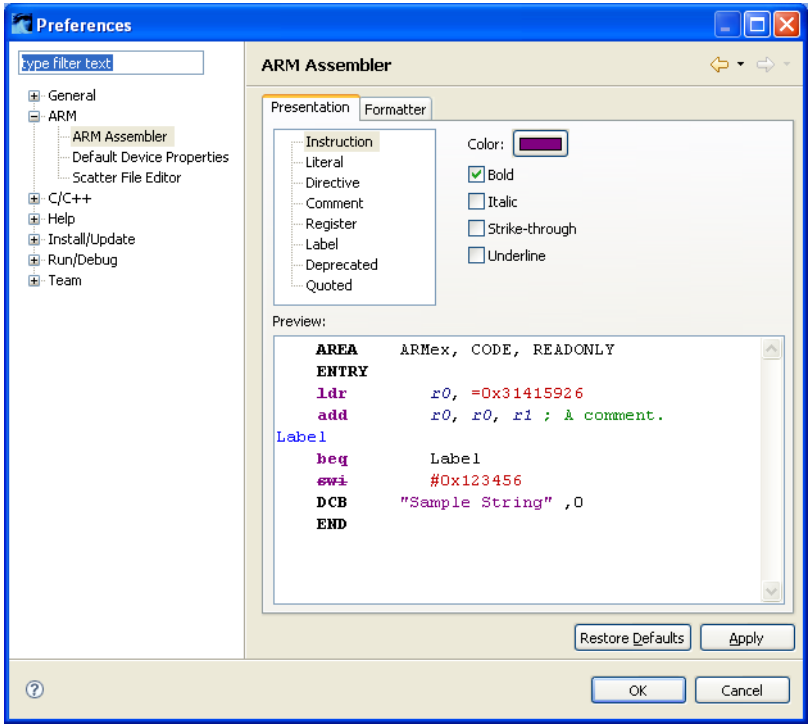

图 2-11 窗口首选项对话框

2.4.2 属性对话框

使用“Properties/属性”对话框可以自定义项目设置，请参阅第2-18 页的图 2-12。 可以通过选择某个项目，然后从 **Project/项目**菜单中选择 **Properties.../属性...** 来访问此对话框。 对自定义设置所做的更改保存在工作区中的项目文件夹中。 您也可以自定义单个文件的 C/C++ 属性，例如，如果希望在生成过程中将特定编译器选项应用于文件。 有关详细信息，请参阅第 4 章 *配置生成和编译工具*。

注意

如果为单个源文件指定了不同的选项，则这些选项会覆盖在适用于所有相关源文件的项目配置面板中指定的选项。

项目的属性层次结构树内容如下：

Resource/资源

显示资源位置、修改状态和文件类型。

Builders/生成器

控制对所选项目可用的生成器。

C/C++ Build/C/C++ 生成

控制活动配置的环境、生成和工具链设置。

C/C++ General/C/C++ 常规

控制文档、文件类型、索引器和路径/符号设置。

Project References/项目引用

控制项目相关性。

Run/Debug Settings/运行/调试设置

控制所选资源的启动配置。

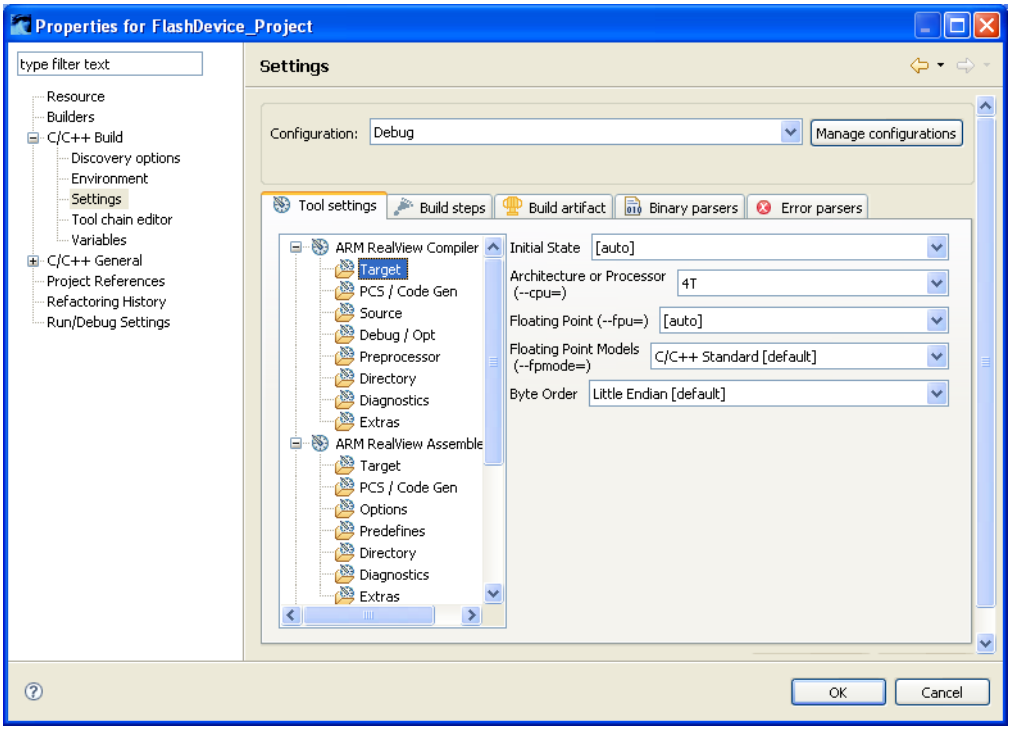

图 2-12 项目属性对话框

2.5 生成

生成是编译和链接源文件以生成输出文件的过程。 生成可以应用于特定项目集或整个工作区。 不能生成单个文件或子文件夹。 有关详细信息，请参阅第 4 章 *配置生成和编译工具*。

Workbench 提供增量生成，这种生成将所选生成配置应用于自上次生成以来更改过的资源。 另一种生成是完整生成，这种生成将所选生成配置应用于所有资源，放弃以前的所有生成状态。

自动

这是增量生成，对整个工作区进行操作，可以在保存资源时自动运行。 只有在项目行为对话框中选择 **Build on resource save (Auto build)/保存资源时生成(自动生成)**，才能为每个项目都启用此行为，请参阅图 2-13。 缺省情况下，任何项目都未选择此行为。

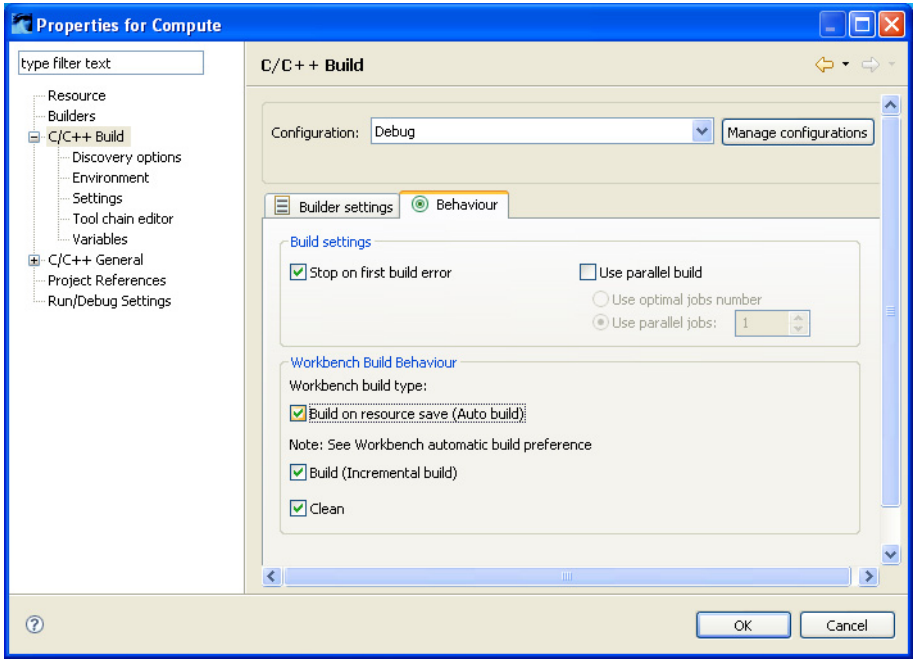

图 2-13 Workbench 生成行为

此外，还必须确保从 **Project/项目**菜单中选择 **Build Automatically/自动生成**。 缺省情况下，此菜单选项为选中状态。

手动

这是增量生成，对整个工作区中选择了 **Build (Incremental build)/生成(增量生成)** 的项目进行操作，请参阅第2-19 页的图 2-13。 缺省情况下，所有项目都选择此行为。

从“Project/项目”菜单中选择 **Build All/全部生成**或 **Project/项目 → Build Project/生成项目**可以运行增量生成。

——注意——

手动生成不会在运行前进行保存，因此必须在选择此选项前保存所有相关文件！ 若要在生成前自动保存，可以通过从“Window/窗口”菜单中选择 **Preferences.../首选项... → General/常规 → Workspace/工作区**来更改缺省设置。

Clean/完整

此选项放弃以前的所有生成状态（包括所选项目中的目标文件和映像）。 完整生成后的下一次自动或手动生成会将所选生成配置应用于所有资源。

从 **Project/项目**菜单中选择 **Clean.../完整...** 对整个工作区或特定项目运行完整生成。 此外，还必须确保在“Preferences/首选项”对话框的 **C/C++ Build/C/C++ 生成 → Behaviour/行为**选项卡中选中 **Clean/完整**，请参阅第2-19 页的图 2-13。 缺省情况下，所有项目都选择此行为。

生成顺序这一功能用于创建项目间相关文件并定义特定生成顺序。 例如，某个映像可能需要以特定顺序生成多个目标文件。 为此，必须将目标文件拆分为较小的不同项目，在一个较大项目中引用这些项目可确保在生成较大项目之前生成这些较小项目。 生成顺序还可以应用于被引用的项目。

有关生成顺序和项目引用的详细信息，请参阅：

- 第2-16 页的 *“Preferences/首选项”对话框*
- 第2-17 页的*属性对话框*

2.6 导入和导出

资源必须在 Workbench 中的项目中，才能用于生成。 如果要在某个项目中使用文件系统中的现有资源，建议使用“Import/导入”向导。 为此，请从 **File/文件**菜单中选择 **Import.../导入...**。

如果要在 Workbench 外部使用资源，建议使用“Export/导出”向导。 为此，请从 **File/文件**菜单中选择 **Export.../导出...**。

导入和导出向导中提供了几个选项。 Workbench 支持以下选项：

General/常规 此选项用于导入和导出以下内容：

- 压缩 zip 文件
- 包含断点设置的文件
- 完整项目
- 所选源文件和项目子文件夹
- 包含 Workbench 首选项设置的文件

C/C++ 此选项用于导入以下内容：

- C/C++ 可执行文件
- CodeWarrior 项目

Flash programmer/Flash 编程器

此选项用于导入和导出以下内容：

- 将 Flash 映像文件导入 Flash 项目
- 导出供 RealView Debugger 使用的板配置
- 导出供 RealView Debugger 使用的 Flash 设备

有关此处未列出的其他选项的信息，请参阅动态帮助。

2.6.1 导入

“Import/导入”向导可以用于导入完整项目、源文件和项目子文件夹以及断点和首选项设置。 从 **File/文件**菜单中选择 **Import.../导入...**。

本节重点介绍项目、源文件和项目子文件夹。

若要从压缩 zip 文件或文件系统中的外部文件夹导入完整项目，必须使用“Existing Projects into Workspace/将现有项目导入工作区”或“CodeWarrior Project exported as XML/导出为 XML 的 CodeWarrior 项目”向导。 这样可确保将相关 Workbench 文件也导入工作区。 有关详细信息，请参阅第 3 章 *使用项目*。

使用“Archive File/压缩文件”或“File System/文件系统”向导可以导入单个源文件和项目子文件夹。 选择这两个选项，都会出现一个对话框，类似于图 2-14 所示的示例。 使用所提供的选项，可以选择所需资源，指定相关选项、文件名和目标路径。

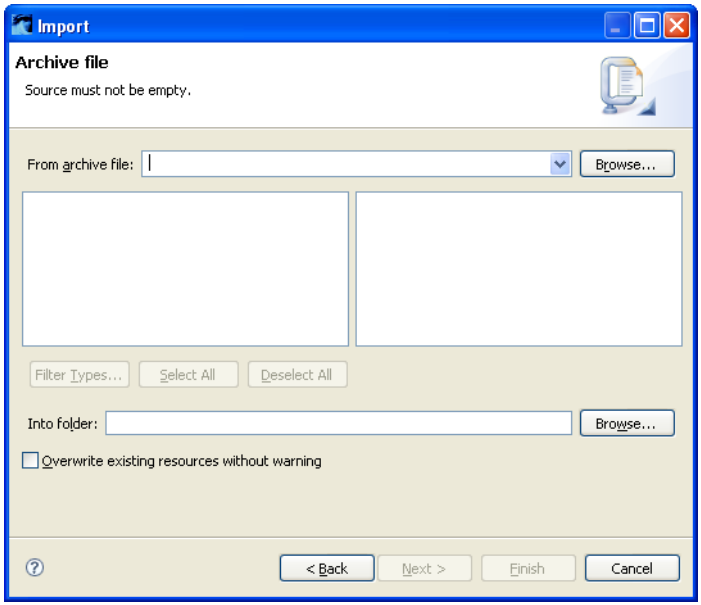

图 2-14 导入向导的典型示例

除“Existing Projects into Workspace/将现有项目导入工作区”之外，在使用“Import/导入”向导时，会将文件和文件夹复制到工作区中。 若要创建指向外部文件或项目子文件夹的链接，必须使用“New File/新建文件”或“New Folder/新建文件夹”向导，有关详细信息，请参阅第2-7 页的*链接资源*。

2.6.2 导出

“Export/导出”向导可以用于导出完整项目、源文件和项目子文件夹以及断点和首选项设置。 从 **File/文件**菜单中选择 **Export.../导出...**。

本节重点介绍项目、源文件和项目子文件夹。

导出完整项目、源文件或项目子文件夹可使用相同的过程。 如果要创建 zip 文件，可以使用“Archive File/压缩文件”向导，也可以使用“File System/文件系统”向导。 选择这两个选项，都会出现一个对话框，类似于图 2-15 所示的示例。 使用所提供的选项，可以选择所需资源，指定相关选项、文件名和目标路径。

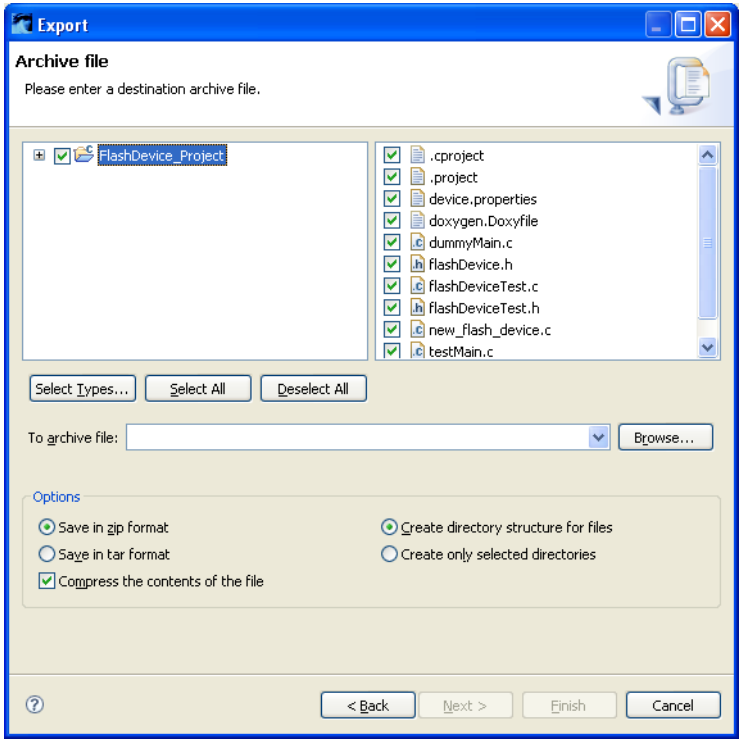

图 2-15 导出向导的典型示例

2.7 获得帮助

Workbench 提供了可从 Workbench 进行访问的以下帮助：

- 相关内容的动态帮助
- 提供特定任务指南的备忘单
- 安装 Workbench 的软件更新
- 访问 ARM 网站
- 教程

2.7.1 动态帮助

若要访问特定 Workbench 功能的动态帮助，必须：

1. 单击要使用的功能的可编辑字段。
2. 单击问号图标。

所选功能的动态帮助显示在“Related Topics/相关主题”视图顶部的“About/关于”面板中。

注意

其他搜索结果列在“Related Topics/相关主题”视图底部的“Dynamic Help/动态帮助”面板中。

另一种查看动态帮助的方式是使用“Help/帮助”窗口：

1. 从 **Help/帮助**菜单中选择 **Help Contents/帮助目录**。
2. 在“Contents/目录”框架中，选择 **ARM Workbench IDE Dynamic Help/ARM Workbench IDE 动态帮助**，然后选择要获得帮助的插件。

2.7.2 ARM 特定备忘单

备忘单是一些实用的示例，可用于指导您完成特定任务。“Cheat Sheets/备忘单”视图中列出了任务的每个步骤，突出显示并展开当前步骤。您必须依次执行每个步骤才能完成任务。

使用备忘单：

1. 从 **Help/帮助**菜单中选择 **Cheat Sheet.../备忘单...**。
2. 从列表中选择备忘单，或使用 **Browse.../浏览...** 从文件中进行选择，请参阅第2-25 页的图 2-16。单击 **OK/确定**。

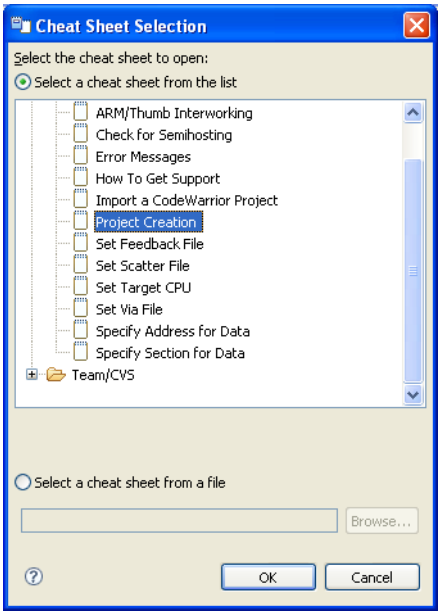

图 2-16 选择备忘单

3. 单击“Introduction/简介”步骤中的 **Click to Begin/单击开始**链接启动任务。 如果多次打开某个备忘单，则“Introduction/简介”步骤中的链接会将文本更改为 **Click to Restart/单击重新开始**。

4. 按照操作说明一步步执行。 完成一条操作说明后，单击 **Click to Complete/单击完成**链接可前进到下一条操作说明。 某些操作说明可能有 **Click to Perform/单击执行**链接，供您在希望由备忘单自动执行该操作时使用。

执行完“Cheat Sheets/备忘单”视图中所列的所有步骤后，就完成了任务。 备忘单左边距中所示的勾号图标表示已完成，请参阅第2-26 页的图 2-17。

第2-26 页的图 2-17 是备忘单的典型示例。 步骤一和二显示为已完成，步骤三（当前步骤）为突出显示和展开状态，已可使用。 步骤三中列出了四条操作说明，依次完成每一条操作说明。 完成步骤三的全部操作时，备忘单会继续往下，填充并显示步骤四中的操作说明。

注意

只有完全完成前一个步骤之后，才会填充后续步骤。

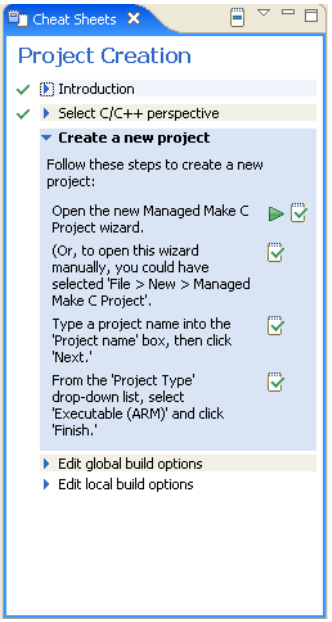

图 2-17 备忘单示例

2.7.3 安装新功能

安装新功能：

1. 选择 **Help/帮助** → **Software Updates/软件更新** → **Find and Install.../查找并安装...**
2. 在“Install/Update/安装/更新”对话框中选择 **Search for new features to install/搜索要安装的新功能**。 单击 **Next/下一步**。
3. 从 **Sites to include in search/要搜索的站点**列表中选择 ARM 远程站点。 单击 **Finish/完成**。
4. 阅读许可协议并接受该协议。 如果不接受许可协议，则无法安装该功能。 单击 **Next/下一步**，然后单击 **Finish/完成**。
5. 在“Verification/验证”对话框中，单击 **Install All/全部安装**。
6. 单击 **Yes/是**重新启动 Workbench 并完成安装。

若要安装更新的功能，请遵循相同步骤，但要在“Install/Update/安装/更新”对话框中选择 **Search for updates of the currently installed features/搜索当前安装的功能的更新**。

注意

可以在“Preferences/首选项”对话框的“Install/Updates/安装/更新”面板中启用自动更新。 为此，请从“Window/窗口”菜单中选择 **Preferences.../首选项...**。

2.7.4 访问 ARM 网站

Workbench 提供了到 ARM 网站的以下外部链接：

- ARM 文档
- ARM 自助论坛
- ARM 技术支持下载
- ARM 开发工具常见问题
- ARM 技术支持

若要访问这些链接，请从“Help/帮助”菜单中选择 **ARM on the Web/ARM 网站**。

2.7.5 教程

ARM 网站提供一些动画教程，用于演示 Workbench 中的 ARM 插件的最常见用法。 访问教程：

1. 从“Help/帮助”菜单中选择 **Help Contents/帮助目录**。
2. 从“Contents/目录”框架中选择 **ARM Workbench IDE Dynamic Help/ARM Workbench IDE 动态帮助**。
3. 选择 **ARM Tutorials/ARM 教程**。

2.8 使用限制

本节列出了使用 Workbench 时适用的特定限制和注意事项。

组织项目

项目源文件的建议结构是在项目文件夹或子文件夹中创建这些文件。 如果源文件在项目之上的文件夹中创建的，则会创建绝对链接。

打开现有的 Eclipse 项目

必须使用导入向导。 有关详细信息，请参阅第3-9 页的*导入现有的 Eclipse 项目*。

CodeWarrior 子工程

不支持 CodeWarrior 子工程。 您必须逐个导入每个工程。 有关详细信息，请参阅第3-11 页的*导入现有的 CodeWarrior 项目*。

工程间的相关性

不支持嵌套工程。 每个工程都必须组织为单独的实体。 通过引用工作区中的其他项目，可以设置项目间相关性。 从“Project/项目”菜单中选择 **Properties/属性 → Project References/项目引用**可手动添加引用。

链接顺序 在 Workbench 中，不能在同一项目内指定目标文件的链接顺序。 一种解决方法是将目标文件拆分为不同的项目，这样即可指定项目生成顺序。 从“Window/窗口”菜单中选择 **Preferences.../首选项... → General/常规 → Workspace/工作区 → Build Order/生成顺序**。

恢复缺省值

恢复项目的缺省值将丢弃不属于项目类型的所有信息。 在“ARM New Project Wizard/ARM 新建项目向导”中更改的所有设置都会丢失。

使用现有目标配置启动 RealView Debugger

在 Workbench 中使用具有现有目标配置的项目启动 RealView Debugger 时，可能会显示 RealView Debugger 对话框，如第2-29 页的图 2-18 所示。 单击 **No/否**可在 Workbench 中使用目标配置。

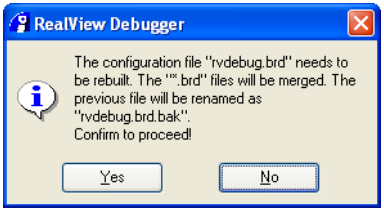

图 2-18 RealView Debugger 消息

第 3 章
使用项目

可以使用 Workbench 为 ARM 目标创建项目。 项目是工作区的顶层文件夹，它包含相关的文件和子文件夹。 在添加新文件或导入现有文件之前，工作区中必须包含项目。

——注意——

RealView 项目向导集 C 项目和 C++ 项目的功能于一身。 这允许您在同一个项目中生成 `.c`、`.cpp` 和 `.s` 文件。

本章内容如下：

3.1 关于 ARM 项目类型

本节介绍有关 Workbench 所提供的各种 ARM 项目类型的信息。 从“File/文件”菜单中选择 **New/新建 → RealView Project/ Real View 项目**，可以选择一种项目类型和关联的工具链，请参阅图 3-1。

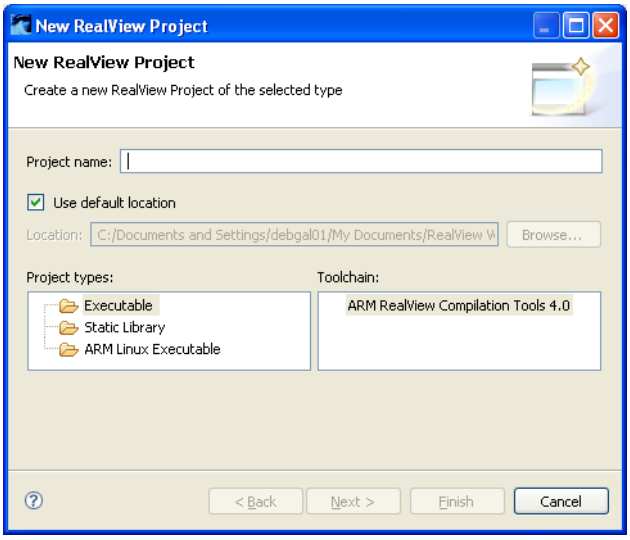

图 3-1 安装的工具链的 ARM 项目类型

注意

如果安装了多个产品，向导中只会列出最新的工具链。

您可以使用“Project Converters/项目转换器”对话框更新早期的项目，这样就可以使用最新的工具链。 请注意，如果还原为一个早期版本，可能会失去工具链功能。 若要访问“Project Converters/项目转换器”对话框，请右击项目，在上下文菜单中选择 **Convert To…/转换为…**，请参阅图 3-2。

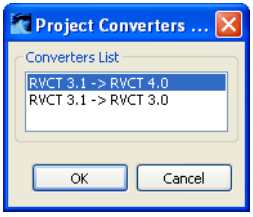

图 3-2 Project converters/项目转换器

3.1.1 Executable/可执行文件

使用“Executable/可执行文件”项目类型可由 ARM 和 Thumb® 代码创建可执行的 ELF 映像。 可执行的项目使用：

- ARM 编译器 (armcc)，用于在 ARM 或 Thumb 状态下编译 C 和 C++ 源文件。
- ARM 汇编程序 (armasm)，用于汇编文件扩展名为 .s 的文件。
- ARM 链接器 (armlink)，用于链接可执行的 ELF 映像。
- RealView Debugger，用于调试和运行由项目生成输出的可执行映像。

您也可以将 Workbench 配置为调用 ARM fromelf 实用程序来基于可执行的 ELF 映像创建二进制文件。 请参阅第4-6 页的*使用 ARM fromelf 实用程序*。

3.1.2 Static library

使用“Static Library/静态库”项目类型可生成 ELF 对象格式成员库。 此项目类型与“Executable/可执行文件”类似。 静态库项目使用 ARM 库管理程序 (armar) 输出对象库，而不是使用 ARM 链接器 (armlink) 输出可执行的 ELF 映像。

——注意——
只有将独立的库文件链接到某个映像后，才能对其进行调试或运行。

3.1.3 Linux 的 ARM 项目类型

Linux 的 ARM 项目需要安装 CodeSourcery ARM GCC。 有关详细信息，请参阅 ARM 网站上相关的应用程序说明。

Linux 可执行文件

使用 Linux“Executable/可执行文件”项目类型可为 ARM 和 Thumb 代码创建可执行的 ELF 映像。 Linux“Executable/可执行文件”项目使用：

- ARM 编译器 (armcc)，用于在 ARM 或 Thumb 状态下编译 C 和 C++ 源文件。
- ARM 汇编器 (armasm)，用于汇编文件扩展名为 .s 的文件。
- ARM 链接器 (armlink)，用于链接可执行的 ELF 映像。

- RealView Debugger，用于调试和运行由项目生成输出的可执行映像。

3.1.4 生成配置

缺省情况下，新建项目向导提供两个单独的生成配置：

Debug/调试 调试目标配置为生成完全可调试的输出二进制文件，但优化程度较低。 它将编译器优化设置配置为最低（级别 0），以便为代码开发提供理想的调试视图。

Release/发布 发布目标配置为生成高度优化的输出二进制文件，但调试视图的质量较差。 它将编译器优化设置配置为高（级别 2）。 调试信息仍包括在生成的映像中。

在所有的新项目中，Debug 配置都自动设置为活动配置。 它可以在 C/C++ Build 配置面板中更改。 有关信息，请参阅第 4 章 *配置生成和编译工具*。

3.1.5 项目模板套件

新建项目向导提供模板列表，包含特定板配置和 ARM 处理器的缺省设置。 例如，ARM7TDMI® 处理器。

缺省情况下，选择 ARM7TDMI 处理器，但是如果您已知目标的正确配置，可以从提供的列表中选择特定目标。 选择正确的目标可确保自动设置所有的缺省设置。

有关详细信息，请参阅第3-5 页的*创建新 RealView 项目*。

3.2 创建新 RealView 项目

本节介绍如何使用 RealView 项目向导创建新项目。

创建新 RealView 项目：

1. 从“File/文件”菜单中选择 **New/新建 → RealView Project/RealView 项目**。
2. 输入项目名称，然后选择项目类型，请参阅图 3-3。 使 **Use Default location/使用缺省位置**选项保持为选中状态，以便在显示的缺省文件夹中创建项目。 或者，取消选择此选项，然后浏览至所需的项目文件夹。 单击 **Next/下一步**。

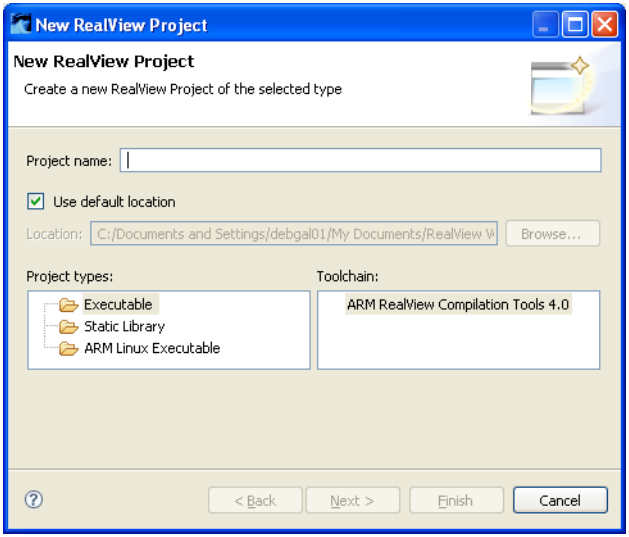

图 3-3 为项目命名

3. 在配置对话框中，使 **Debug/调试**和 **Release/发布**选项保持为选中状态，以便更改其他配置的项目设置。 单击 **Next/下一步**。
4. 第3-6 页的图 3-4 是“Target/目标”对话框，其中的模板列表包含特定目标的缺省设置。 根据需要选择模板。 单击 **Next/下一步**。

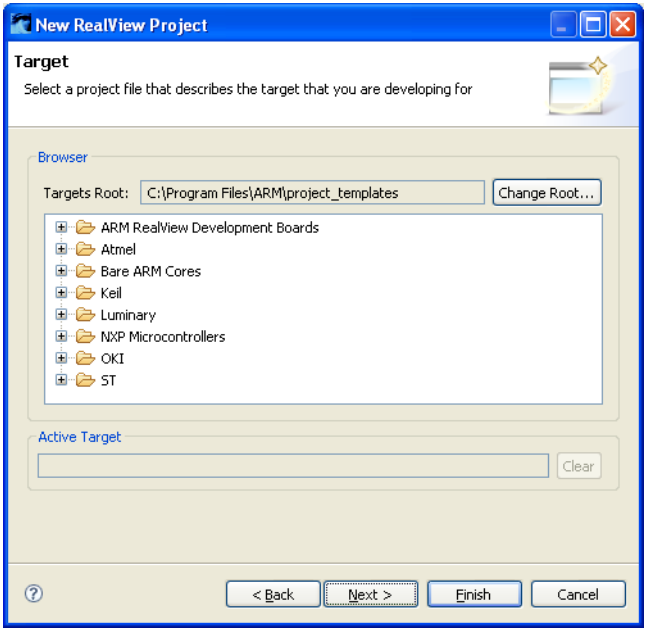

图 3-4 项目模板选择

5. 图 3-5 显示的是“Core Settings/核心设置”对话框。 选择 **Architecture or Processor (--cpu=)/体系结构或处理器(--cpu=)**、**Floating Point (--fpu=)/浮点(--fpu)** 和 **Byte Order/字节顺序**。 单击 **Next/下一步**。

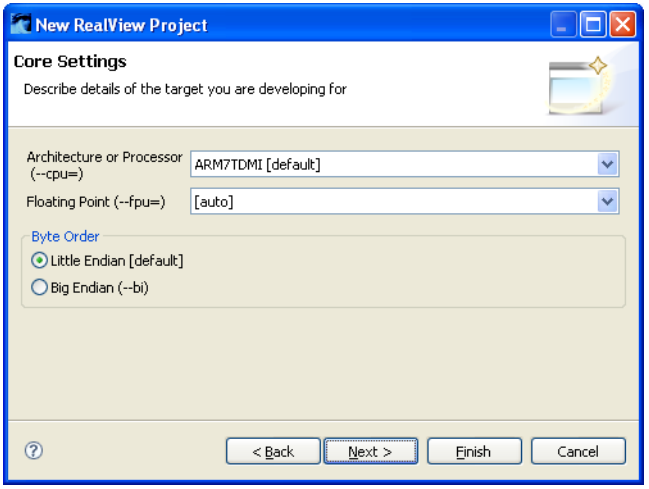

图 3-5 优化目标设置

6. 图 3-6 显示的是“Language Settings/语言设置”面板。 根据需要选择 **Initial State/初始状态**和 **Source Language/源语言**。 使用缺省设置 **[auto]/[自动]** 可让编译器自动为项目选择各选项。 单击 **Next/下一步**。

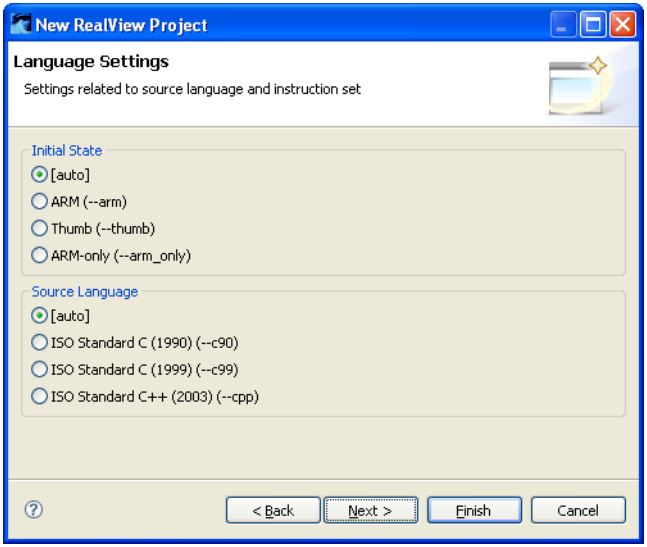

图 3-6 语言设置

7. 在 Linux 项目中，“Linux Target Configuration/Linux 目标配置”对话框提供了更改缺省配置参数的选项。 根据需要进行选择，然后单击 **Finish/完成**创建新项目。 对于其他项目，跳过此步骤。

8. 第3-8 页的图 3-7 显示的是“Examples/示例”对话框，其中的示例包含可用作项目起点的代码。 如果选择缺省选项 **None/无**，则会创建一个不包括示例代码的空项目。 根据需要进行选择，然后单击 **Finish/完成**创建新项目。

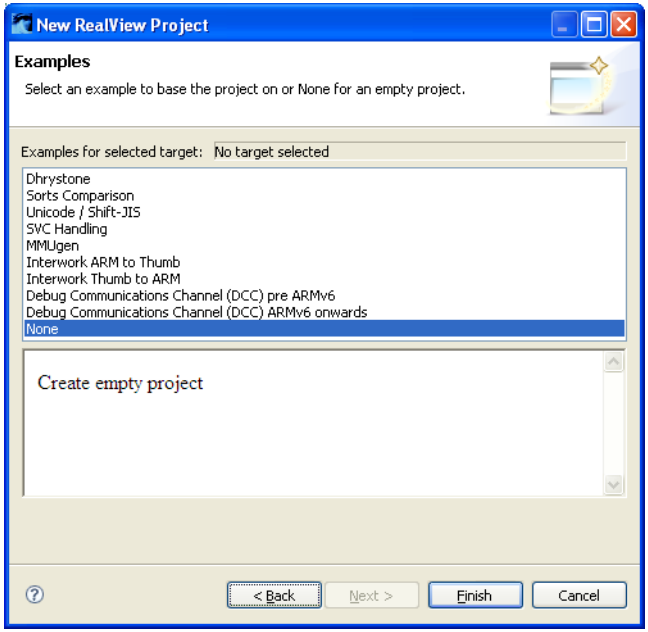

图 3-7 项目示例

项目显示在“Project Explorer/项目资源管理器”视图中。

3.3 导入现有的 Eclipse 项目

本节介绍如何将现有的 Eclipse 项目导入到工作区中。

导入现有的 Eclipse 项目：

1. 从“File/文件”菜单中选择 **Import.../导入...**。
2. 若要导入现有的项目，请选择 **Existing Project into Workspace/将现有项目导入工作区**，如图 3-8 所示。 单击 **Next/下一步**。

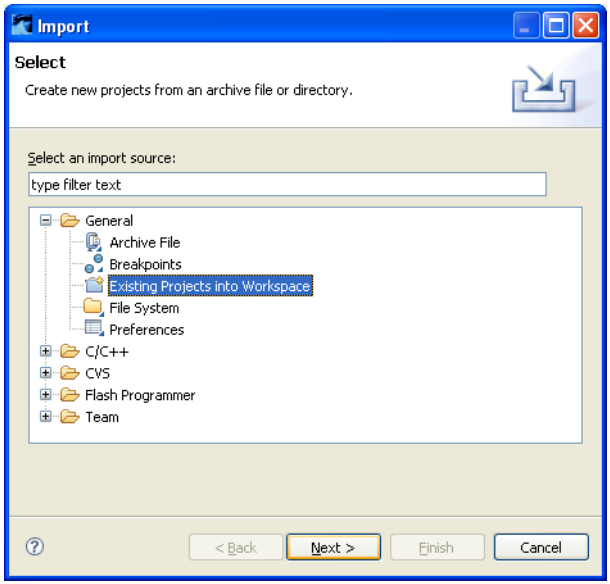

图 3-8 选择导入源类型

3. 单击 **Browse/浏览**选择项目根目录文件夹。 确保已在“Projects/项目”面板中选择了所有需要导入的项目。 根据需要选择 **Copy projects into workspace/将项目复制到工作区中**，或取消选择此选项以创建指向现有项目及关联文件的链接。 请参阅第3-10 页的图 3-9。 单击 **Finish/完成**。

注意

如果现有的项目包含生成系统的旧版本的项目设置，则会显示更新项目的选项。 使用最新版本意味着您可以访问所有最新的工具链功能。

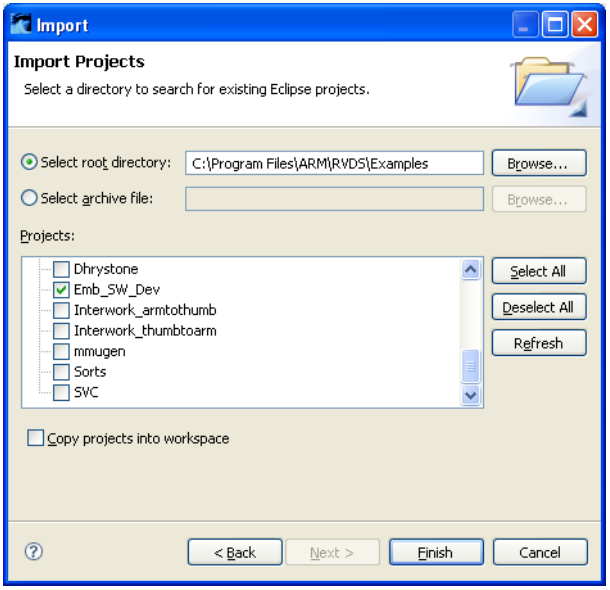

图 3-9 选择要导入的现有 Eclipse 项目

已导入的项目显示在“Project Explorer/项目资源管理器”视图中。

3.4 导入现有的 CodeWarrior 项目

本节介绍如何将使用 RealView Development Suite 创建的项目导入到工作区中。

导入现有的 CodeWarrior 项目：

1. 将 CodeWarrior 项目导出为可扩展标记语言 (XML) 格式（文件扩展名为 `.xml`）。

   注意

   该 `.xml` 文件必须创建在 CodeWarrior 项目文件（文件扩展名为 `.mcp`）所在的目录中。

   有关如何将 CodeWarrior 项目导出为 XML 的详细信息，请参阅《RealView Development Suite CodeWarrior IDE 指南》(*RealView Development Suite CodeWarrior IDE Guide*)。 如果将 CodeWarrior 项目导入到 Workbench 中，则会创建一个新项目，并且相关的工具设置也将传递并应用于新项目。

   CodeWarrior 导入程序存在一些局限性：

   - 不支持子项目。 您必须逐个导入每个项目。
   - 子目标的导入方式与文件相同。 有关详细信息，请参阅第3-15 页的*向项目中添加文件*。
   - 在 Workbench 中没有对应选项的命令行选项包括在每个工具的“Extras/其他”面板中。 有关详细信息，请参阅第4-5 页的*配置 ARM 编译工具*。

2. 从“File/文件”菜单中选择 **Import/导入**。

3. 从“Import/导入”对话框中选择 **CodeWarrior Project exported as XML/导出为 XML 的 CodeWarrior 项目**，请参阅第3-12 页的图 3-10。 单击 **Next/下一步**。

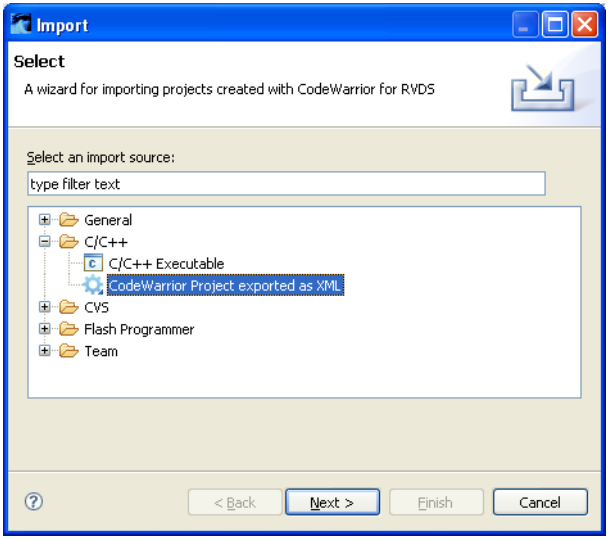

图 3-10 选择 CodeWarrior XML

4. 单击 **Browse…/浏览…** 选择包含 .xml 文件的目录，请参阅图 3-11。

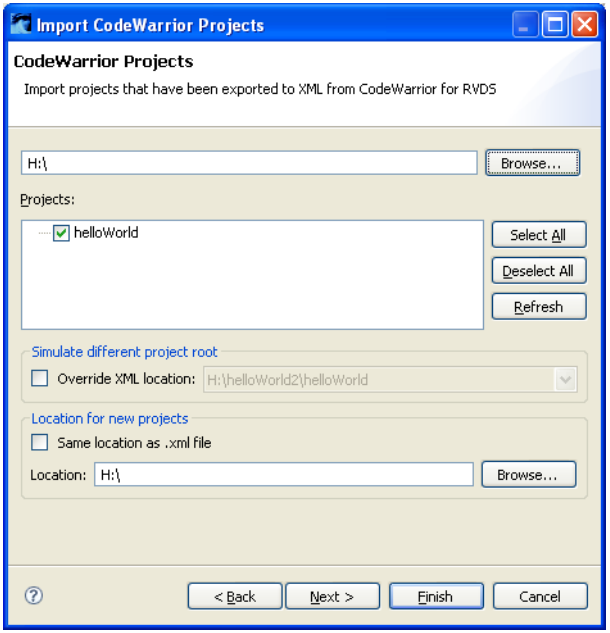

图 3-11 导入 CodeWarrior 项目

5. 缺省情况下，全部可导入的项目都处于选中状态。 如果“Projects/项目”面板显示多个项目，请取消选择不需要导入的项目。

6. 如果源文件位于 CodeWarrior 项目根目录的上层文件夹中，请选择 **Override XML location/覆盖 XML 位置**选项。 这样，您可以选择其他项目根目录，以便导入向导在创建新项目时能够镜像父结构。 图 3-12 中的示例演示的是用作新项目根目录的父结构的项目文件夹。

注意

如果不使用该选项，则必须使用绝对路径才能链接项目根目录之外的资源。

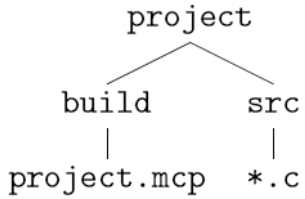

图 3-12 项目文件之上的源文件夹的结构

7. 在“Location/位置”字段中，指定在当前工作区创建新项目的缺省路径，或单击 **Browse.../浏览...** 按钮选择其他位置。 单击 **Next/下一步**。

注意

如果选择 **Same location as .xml file/与 .xml 文件的位置相同**，新项目将合并到现有的 Code Warrior 文件中。

8. 选择 **Filter include paths/筛选包含路径**选项，最小化命令行的包含路径选项，请参阅第3-14 页的图 3-13。 只有包含以指定扩展名结尾的文件的路径才会添加到命令行选项中。 扩展名必须表示为用逗号分隔的列表。 例如： *h,inc*。

注意

如果不使用该选项，则项目中的所有文件夹都添加为显式包含的，这样会添加太多内容。

9. 链接资源的位置可以相对于路径变量指定，请参阅第3-14 页的图 3-13。 这样，不必镜像相同的文件结构，就可以共享包含链接资源的项目。 如果没有指定任何路径变量，则 **Link files relative to variable/相对于变量链接文件**选项为禁用状态。 单击 **Finish/完成**创建新项目。

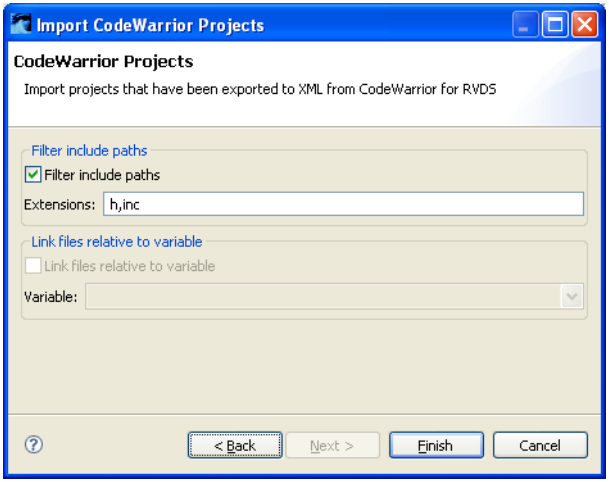

图 3-13 筛选包含路径和路径变量。

注意

若要添加新的路径变量，请使用“Preferences/首选项”对话框的 **General/常规 → Workspace/工作区 → Linked Resources/链接资源**选项。

如果使用路径变量导入项目，向导会自动更新“Preferences/首选项”对话框的 **C/C++ → CDT build variables/CDT 生成变量**选项，以与 **General/常规 → Workspace/工作区 → Linked Resources/链接资源**选项保持一致。 两个选项必须彼此同步。 如果随后移动了用路径变量链接的资源，则必须更新这两个选项。

已导入的项目显示在“Project Explorer/项目资源管理器”视图中。

3.5 向项目中添加文件

本节介绍如何向项目添加空的源文件。

向项目添加新文件：

1. 从“File/文件”菜单中选择 **New/新建** → **File/文件**。
2. 从“New File/新建文件”对话框中选择项目文件夹或子文件夹，在 **File name/文件名**文本框中输入具有相关扩展名的文件名。 请参阅图 3-14。

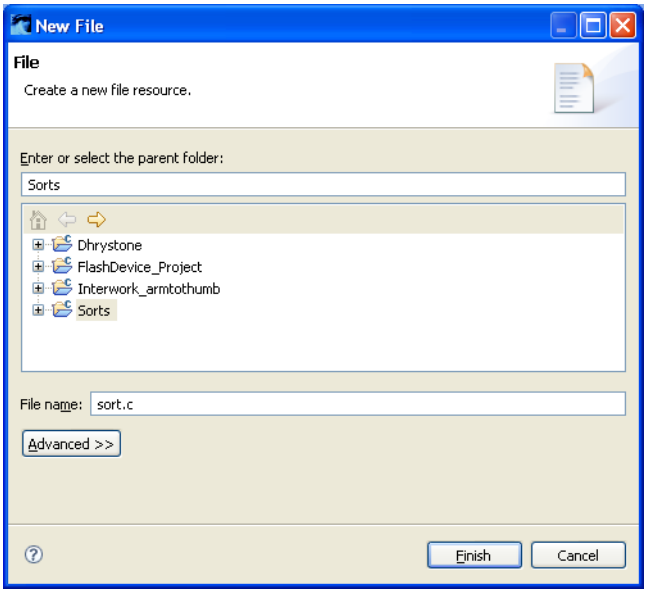

图 3-14 为文件命名

3. 单击 **Finish/完成**。

   新的源文件显示在“Project Explorer/项目资源管理器”视图中。

注意

- 利用文件资源管理器，您还可以在项目文件夹中创建文件或直接将文件拖到项目文件夹中。 若要在 Workbench 中更新视图，请在“Project Explorer/项目资源管理器”视图中单击相关项目，然后从 **File/文件**菜单中选择 **Refresh/刷新**。
- 也可以在 Workbench 的“Project Explorer/项目资源管理器”视图中将文件直接拖放到项目文件夹中。

3.6 向项目中添加库

本节介绍如何向项目中添加库。

向项目添加新库：

1. 在“Project Explorer/项目资源管理器”视图中选择项目。
2. 在“Project/项目”菜单中选择 **Properties/属性**。
3. 在“Properties/属性”对话框中选择 **C/C++ Build/C/C++ 生成 → Settings/设置**。
4. 在“Tool Settings/工具设置”选项卡中，从 ARM 链接器设置中选择 **Directory/目录**。
5. 在“Library/库”面板中单击“Add/添加”按钮，打开“Add file path/添加文件路径”对话框。
6. 单击 **Workspace.../工作区...** 按钮或 **File system.../文件系统...** 按钮选择库文件。
7. 单击 **OK/确定**关闭对话框。

   现在，ARM 链接器就可以使用该库文件了。

——注意——

如果生成的文件路径包含空格，链接器会产生一条错误消息。 通过删除“Library/库”面板中的输入内容中的引号，可以解决这一已知问题。 单击 **Edit/编辑**按钮修改文件路径。

第 4 章
配置生成和编译工具

本章介绍如何使用“Properties/属性”对话框的“C/C++ Build/C/C++ 生成”配置设置来配置生成。 这些设置确定 ARM® RealView® 编译工具生成 ARM 可执行映像或库的方式。

注意

生成配置设置可以应用于各个文件和完整的项目。 若要访问“Properties/属性”对话框，请在“Project Explorer/项目资源管理器”视图中单击某个项目文件夹或文件，然后从 **File/文件**菜单中选择 **Properties/属性**。

本章分为以下几节：

4.1 访问 ARM 项目的生成属性

使用“C/C++ Build/C/C++ 生成”配置面板可以在项目中为特定生成配置设置编译工具。 本节介绍如何在项目中访问会影响所有源文件的生成配置面板：

1. 在“Project Explorer/项目资源管理器”视图中选择一个项目。 如果没有项目可用，则添加一个项目；有关详细信息，请参阅第 3 章 *使用项目*。
2. 在 **Project/项目**菜单中选择 **Properties/属性**。
3. 在“Properties/属性”对话框中选择 **C/C++ Build/C/C++ 生成 → Settings/设置**。
4. “Active configuration/活动配置”面板显示当前的项目类型和配置。 使用配置下拉菜单选择：
   - **Release/发布**。 这会显示用于发布生成的编译工具设置。
   - **Debug/调试**。 这会显示用于调试生成的编译工具设置。

   ——注意——
   通过在“Properties/属性”对话框的“Active configuration/活动配置”面板中选择 **Manage/管理**，可以创建新的生成配置。
5. 项目可用的编译工具及其相应的配置面板显示在 **Tool Settings/工具设置**选项卡中。 请参阅第4-3 页的图 4-1。

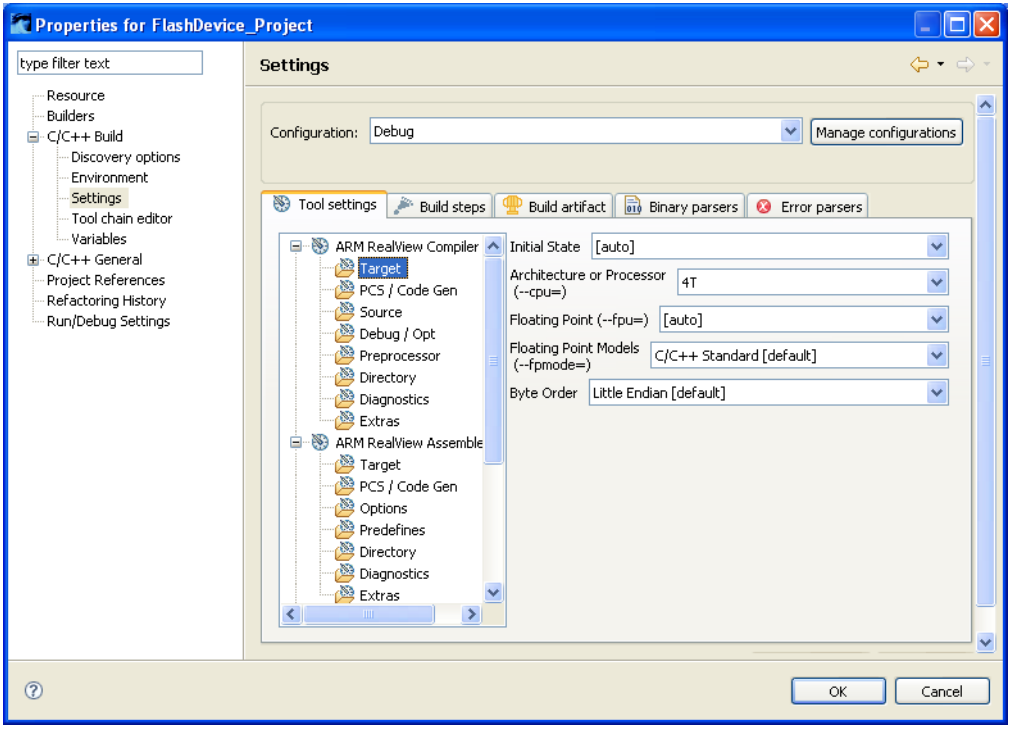

图 4-1 ARM 项目的生成配置面板

4.2 访问特定文件的生成属性

可以为每个源文件设置不同的 ARM 编译器和汇编器。

注意

如果为单个源文件指定了不同的选项，则这些选项会覆盖在项目配置面板中指定的适用于所有源文件的选项。

本节介绍如何访问项目中任意特定源文件的配置面板：

1. 在“Project Explorer/项目资源管理器”视图中选择源文件，然后从“File/文件”菜单选择 **Properties/属性**。
2. 在“Properties/属性”对话框中选择 **C/C++ Build/C/C++ 生成 → Settings/设置**。
3. 这会显示特定于所选文件的配置设置。 编译工具显示在 **Tool Settings/工具设置**选项卡中。 如果源文件的扩展名为 `.s`，则 **Tool Settings/工具设置**选项卡显示 ARM 汇编器设置。 如果源文件的扩展名为 `.c` 或 `.cpp`，则 **Tool Settings/工具设置**选项卡显示 ARM 编译器设置。

4.3 配置 ARM 编译工具

为便于查看和设置选项，为每个 ARM 编译工具都提供了多个面板，例如编译器 (armcc)、汇编器 (armasm)、链接器 (armlink) 和库管理程序 (armar)。 若要访问这些面板，请参阅第4-2 页的*访问 ARM 项目的生成属性*。

注意

- 只有可执行项目类型才会显示 ARM 链接器面板。
- 只有静态库项目类型才会显示 ARM 库管理程序面板。

有关详细信息，请参阅第3-2 页的*关于 ARM 项目类型*。

有关这些面板中显示的每个选项的介绍，请访问动态帮助。 有关详细信息，请参阅第2-24 页的*动态帮助*。 每个工具的父面板都包含一个“All options/所有选项”字段，其中显示可为该工具设置的所有选项。 使用编译工具时，此选项列表会传递给编译工具。

注意

- “All options/所有选项”字段还包含一些选项，可使 RealView 工具在 Workbench 中正常运行。
- 在使用编译工具时，设置为编译工具缺省值的选项不是必需的，因此不包含在“All options/所有选项”字段中。 例如，如果将编译器优化级别设置为缺省值（级别 2），则 -O2 选项不会出现在“All options/所有选项”字段中。

每个工具的 **Tool Settings/工具设置**选项卡中都有一个“Extras/其他”面板，在该面板中，可以指定其他面板中无法设置的选项。 使用“Extras/其他”面板设置的选项将覆盖在其他面板中设置的选项。

4.4 使用 ARM fromelf 实用程序

ARM `fromelf` 实用程序可将 ARM 链接器生成的可执行和可链接格式 (ELF) 映像文件转换为适用于 ROM 工具的其他格式，以直接加载到内存中。 有关使用 `fromelf` 的详细信息（包括有关输出格式的信息），请参阅《RealView 编译工具实用程序指南》。 将 Workbench 配置为根据可执行 ELF 映像创建纯二进制文件：

1. 转到项目的生成配置面板。 请参阅第4-2 页的*访问 ARM 项目的生成属性*。
2. 在 **C/C++Build/C/C++ 生成 → Settings/设置**配置面板中，单击 **Build Steps/生成步骤**选项卡。
3. 在生成后期步骤的“Command/命令”字段中，输入 `fromelf --bin --output=output.bin` *`inputfile`*，其中 *`inputfile`* 是 ELF 映像的名称。 请参阅图 4-2。

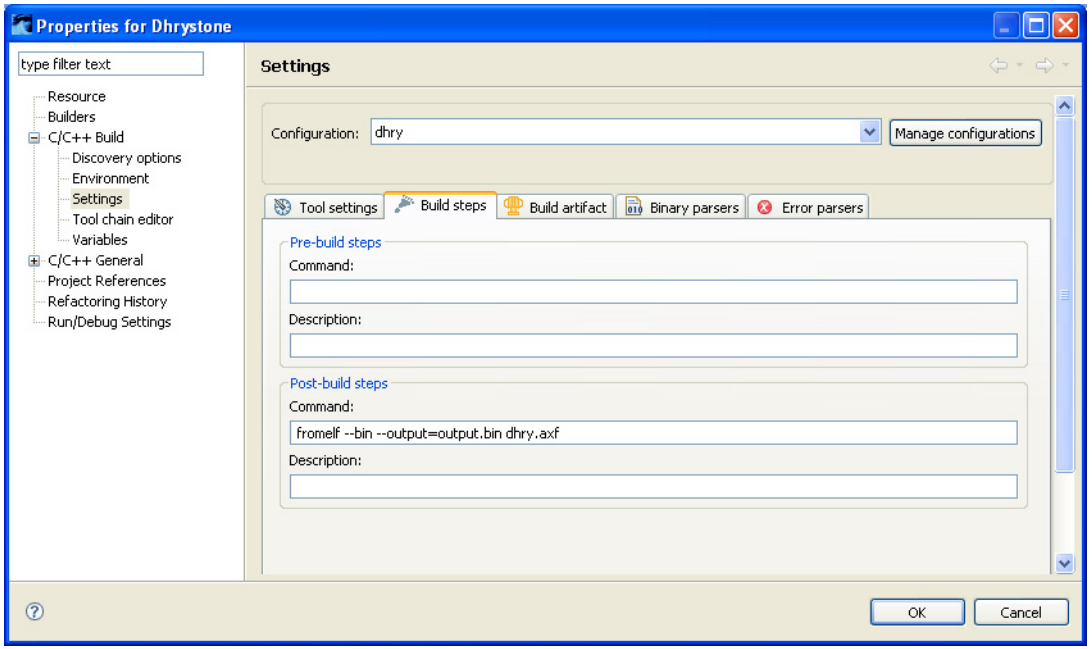

图 4-2 在 Workbench 中调用 `fromelf`

4. 该二进制文件是在生成项目时创建的，以 `output.bin` 形式保存在活动生成配置的子文件夹中。

4.4.1 反汇编代码

使用 ARM `fromelf` 实用程序也可以反汇编 ELF 目标文件并显示各种信息。 反汇编目标文件：

1. 在“Project Explorer/项目资源管理器”视图中展开项目，以显示目标文件。
2. 在“Project Explorer/项目资源管理器”视图中右击目标文件，然后选择 **OpenWith/打开方式 → Elf Content Editor/Elf 内容编辑器**。 或者，也可以双击目标文件。 经过反汇编的目标文件显示在缺省编辑器视图中。 有关详细信息，请参阅第5-19 页的*ELF 内容编辑器*。

4.5 恢复缺省值

使用配置面板（请参阅第4-2 页的*访问 ARM 项目的生成属性*）上的 **Restore Defaults/恢复缺省值**可以重置 **Tool Settings/工具设置**选项卡中显示的所有编译工具的设置。 工具选项将还原为特定于所选项目类型的缺省值。 它只影响活动的生成配置。 例如，如果活动配置是 **Debug/调试**，则只重置调试配置的所有工具设置。 发布配置并不受影响。

注意

- 如果单击项目配置面板（请参阅第4-2 页的*访问 ARM 项目的生成属性*）中的 **Restore Defaults/恢复缺省值**，则不会重置使用特定于文件的配置面板（请参阅第4-4 页的*访问特定文件的生成属性*）设置的任何选项。
- 如果在特定于文件的配置面板中单击 **Restore Defaults/恢复缺省值**，则此面板上的设置会还原为通用配置面板上的设置。 它同样不影响使用项目中其他文件的配置面板设置的任何选项。

第 5 章
使用编辑器

本章介绍在为 ARM 目标开发项目时如何自定义和使用编辑器。

通过更改缺省设置，可以使编辑器与特定文件类型相关联。 在“Preferences/首选项”对话框中，可以查看缺省编辑器关联和其他设置。 有关详细信息，请参阅第2-16 页的 *“Preferences/首选项”对话框*。

在“Project Explorer/项目资源管理器”视图中双击某个文件可调出缺省编辑器。

本章内容如下：

5.1 C/C++ 编辑器

标准 C/C++ 编辑器由 CDT 插件（该插件向 Workbench 提供 C 和 C++ 扩展）提供。 该编辑器在编辑 C/C++ 代码时提供语法突出显示、代码格式设置和内容帮助。 有关详细信息，请参阅动态帮助中的《C/C++ 开发用户指南》。

如果它不是缺省编译器，请在“Project Explorer/项目资源管理器”视图中右击源文件，然后从上下文菜单中选择 **Open With/打开方式 → C/C++ Editor/C/C++ 编辑器**。

5.2 ARM 汇编器编辑器

ARM 汇编器编辑器为 ARM 汇编语言源文件中的标签提供语法突出显示、代码格式设置和内容帮助。 在“Preferences/首选项”对话框中可以更改缺省设置。有关详细信息，请参阅第2-16 页的 *“Preferences/首选项”对话框*。

如果它不是缺省编辑器，请在“Project Explorer/项目资源管理器”视图中右击源文件，然后从上下文菜单中选择 **Open With/打开方式 → ARM Assembler Editor/ARM 汇编器编辑器**。

可以使用的快捷键如下：

表 5-1 ARM 汇编器编辑器快捷键

内容帮助	内容帮助为活动文件中的标签提供自动完整功能。 在输入跳转指令的标签时，键入部分标签，然后使用以下快捷键显示有效自动完成选项的列表： Ctrl+空格键。 使用箭头键选择所需标签，然后按 Enter 键完成术语。 继续键入可忽略自动完成列表。
编辑器焦点	以下选项可更改编辑器焦点： • 大纲视图提供活动文件中所有区域和标签的列表。 单击某个区域或标签可将编辑器的焦点移至所选项的位置。 • 在跳转指令中选择某个标签并按 F3 可将编辑器的焦点移至所选标签的位置。
格式化程序激活	使用 Ctrl+Shift+F 可激活格式化程序设置。
块注释	使用 Ctrl+; 可以启用或禁用块注释。 选择代码块并应用该快捷键可更改注释状态。

5.3 属性编辑器

属性编辑器用于 ARM 汇编器和 C/C++ 文件。 使用属性编辑器可以配置源代码来提供 GUI 组件，这样无需直接编辑代码即可更改特定的值。 控制标记可以嵌入到源代码内的注释块中，经过初始化后，可以实现用于设置和修改变量或 `#defines`（例如硬件初始化）的 GUI 环境。 更新是同步进行的，因此，可以在源代码或 GUI 组件中进行更改。

如果它不是缺省编辑器，请在“Project Explorer/项目资源管理器”视图中右击源文件，然后从上下文菜单中选择 **Open With/打开方式 → Properties Editor/属性编辑器**。

有关使用属性编辑器的详细信息，请参阅：

- *属性编辑器选项卡*
- 第5-5 页的*在注释块中嵌入控制标记*
- 第5-5 页的*关于控制标记*
- 第5-9 页的*使用示例*

5.3.1 属性编辑器选项卡

属性编辑器的底部显示两个选项卡：

Source/源代码

源代码的文本视图，具有语法突出显示和格式设置功能。

必须使用此选项卡才能编辑用于定义 GUI 组件和关联属性的嵌入式控制标记。

Properties/属性

嵌入式 GUI 组件的图形视图。

——注意——

若要启用 GUI 组件，必须激活编辑器选项卡。 有关详细信息，请参阅第5-5 页的*在注释块中嵌入控制标记*。

5.3.2 在注释块中嵌入控制标记

在 ***Source/源代码*** 选项卡中，可以在注释块中嵌入控制标记，从而在 **Properties/属性**选项卡中生成 GUI 组件。表 5-2 列出了可接受的注释标记。

表 5-2 可接受的注释标记

文件扩展名	注释标记
.asm、.inc、.s	;
.c、.cpp、.h	//

注意

不支持 C 样式 /* */ 注释。

如果正确嵌入了控制标记，则可以使用 ***Source/源代码*** 选项卡直接编辑源代码，也可以使用 **Properties/属性**选项卡中的 GUI 组件。 在这两个选项卡之间切换时，它们是彼此同步的。

若要激活选项卡，必须在注释节中包含以下文本行之一：

```
<<< Use Configuration Wizard in Context Menu >>>
```

或

```
<propertieseditor>
```

是否对 GUI 组件结尾进行标记是可选的，但如果进行了标记，则必须在注释节中包含以下文本行之一：

```
<<< end of configuration section >>>
```

或

```
<propertieseditor>
```

在本节结尾处提供了一些代码示例，用于演示属性编辑器的特定用法，请参阅第5-9 页的 *使用示例*。

5.3.3 关于控制标记

控制标记通常应用于紧接注释节的下一项。 使用索引可以修改代码的特定项，也可以忽略不需要通过 GUI 组件修改的其他项。

项可以是向函数提供的参数，也可以是变量或 #define。 项可以划分为几个表示特定位位置的组件。 此外，在使用 # 修饰符标记将组件值合并回相关项之前，可对组件值应用简单计算。 有关详细信息，请参阅第5-7 页的表 5-4。

注意

如果项是大于 2^64 的数字，则行为是未定义的。 这同样适用于在修饰符标记中指定的参数。

控制标记表示单个组件或一组组件。 若要将组件作为一组，请使用开始标记和结束标记。 若要创建 GUI 组件，请使用选项标记。 有关详细信息，请参阅表 5-3 和第5-7 页的表 5-4。

控制标记可以包含用于定义以下内容的其他修饰符标记：

- 范围限制
- 递增步长
- 字符串条目的大小限制
- 数字中的特定位
- 计算修饰符
- 组标题
- 工具提示

控制标记或修饰符标记之后可以有说明，例如：

```
<h> External Bus Interface (EBI)
```

后面没有修饰符标记的数字控制标记使用 <0x0-0xFFFFFFFF> 作为缺省格式。 如果指定了修饰符标记，则格式由下限值确定。

表 5-3 控制标记

控制	说明
<h>	一组组件的开始标记。 仅用于标题。
<e>	一组组件的开始标记。 标题中包含复选框选项，用于启用或禁用后面的组。 例如： **<e>** 代码的下一项包含启用/禁用值。 索引 0 为缺省值。 **<e.4>** 代码的下一项的位 4 包含启用/禁用值。 **<e1.4>** 索引 1 指定代码的第二项的位 4 包含启用/禁用值。
</h> 或 </e>	一组组件的结束标记。

表 5-3 控制标记 （续）

控制	说明
<q>	复选框选项。 例如： **<q>** 代码的下一项包含启用/禁用值。 索引 0 为缺省值。 **<q.4>** 代码的下一项的位 4 包含启用/禁用值。 **<q1.4>** 索引 1 指定代码的第二项的位 4 包含启用/禁用值。
<o>	组合框或数字显示框选项。 例如： **<o>** 代码的下一项包含值。 索引 0 为缺省值。 **<o.4>** 代码的下一项的位 4 包含值。 **<o1.4..6>** 索引 1 指定代码的第二项的位 4 至 6 包含值。
<s>	文本框选项。 大小限制是可选的。 例如： **<s>** 代码的下一项包含字符串。 **<s.10>** 代码的下一项包含大小限制为 10 个字符的字符串。

表 5-4 修饰符标记

修饰符	说明
<i>	工具提示，用于提供有关上一个组件的帮助。 用于开始标记或选项标记。
<0-31>	生成一个包含一组十进制值的数字显示框。 例如，0 - 31 的值。
<0-100:10>	生成一个数字显示框，其中包含一组十进制值，其增量由步长值定义。 例如，0 - 100 的值，其步长为 10。
<0x40-0x1000:0x10>	生成一个数字显示框，其中包含一组十六进制值，其增量由步长值定义。 例如，0x40 - 0x1000 的值，其步长为 0x10。 **注意** 如果未指定任何修饰符标记，则 <0x0-0xFFFFFFFF> 为缺省值。
<0=> *First_Description*	生成一个包含文本说明的组合框。 本示例生成 *First_Description* 作为组合框中的第一项。 值 0 合并回代码中。
<#+1> <#-1> <#*8> <#/3>	在合并回代码之前应用计算，可用计算为： 加法、减法、乘法和除法。

5.3.4 使用限制

应用的限制如下：

- 如果两个选项控制同一个值，则应用最后的更改
- 标记内不允许使用空白字符
- 字符串范围为 0 至 255 个字符
- 位值范围为 0 至 31
- 数字范围为 0 至 2^64
- 如果数字显示框范围大于以下限制值，则会发生错误：
  - 2^32（十六进制）
  - 2^31（十进制）
  - 如果项是大于 2^64 的数字，则行为是未定义的。 这同样适用于在修饰符标记中指定的参数。

5.3.5 使用示例

下面列出的示例是不完整的源代码摘录，其中包含各种嵌入式控制标记和生成的 GUI 组件。

- 第5-10 页的*总线配置示例*
- 第5-12 页的*密码字符串示例*
- 第5-13 页的*数据读取协议示例*
- 第5-14 页的*堆栈配置示例*
- 第5-15 页的*VPBDIV 定义示例*

总线配置示例

示例 5-1 演示了用于修改以下函数的第二个参数的控制标记：

```
WRITE_MEM(0xFFE00000, 0x2034A9)
```

这由 <e1.x> 和 <o1.x> 控制标记中的索引 1 定义。

注意

如果要嵌入用于以下函数的第一个参数的控制标记，则使用索引 0（例如 <e0.x>）：

```
WRITE_MEM(0xFFE00000, 0x2034A9)
```

示例 5-1 总线配置控制标记

```
//C example
//*****************************************************************************
// <propertieseditor>
//*****************************************************************************
// <h> External Bus Interface (EBI)
//   <e1.13> CSR0: Enable Chip Select 0
//     <o1.20..31> BA: Base Address <0x0-0xFFF00000:0x100000> <#/0x100000>
//                 <i> Start Address for Chip Select Signal
//
//     NOTE: Base Address works with bits 20 to 31 of second argument producing
//     spin box with range and step values. Applies calculation before parsing.
//
//     <o1.7..8>   PAGES: Page Size      <0=> 1M Byte    <1=> 4M Bytes
//                                       <2=> 16M Bytes  <3=> 64M Bytes
//                 <i> Selects Active Bits in Base Address
//     <o1.0..1>   DBW: Data Bus Width   <1=> 16-bit     <2=> 8-bit
//     <o1.12>     BAT: Byte Access Type <0=> Byte-write
//                                       <1=> Byte-select
//     <e1.5>      WSE: Enable Wait State Generation
//       <o1.2..4>   NWS: Number of Standard Wait States <1-8> <#-1>
//     </e>
//     <o1.9..11>  TDF: Data Float Output Time <0-7>
//                 <i> Number of Cycles Added after the Transfer
//   </e>
// </h>
WRITE_MEM(0xFFE00000, 0x2034A9);   // EBI_CSR0: Flash
//*****************************************************************************
// </propertieseditor>
//*****************************************************************************
```

图 5-1 显示此源代码生成的 GUI 组件。

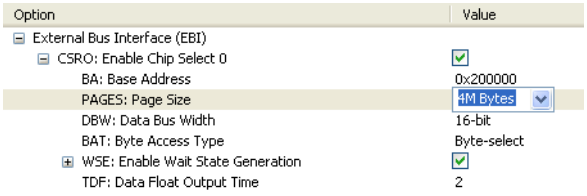

图 5-1 总线配置 GUI 组件

作为一种标示，此示例中嵌入的控制标记在每个说明前都提供了符号名称。图 5-2 列出了这些符号名称并将这些名称与其位位置关联。

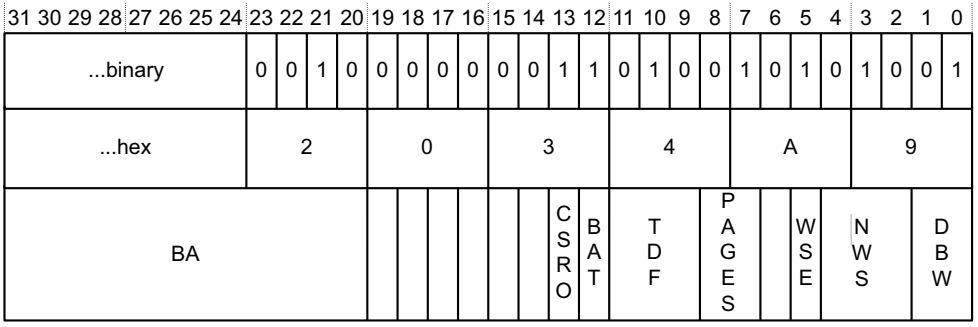

图 5-2 总线配置位位置

密码字符串示例

示例 5-2 中的示例代码生成如图 5-3 所示的 GUI 组件。

示例 5-2 密码字符串控制标记

```
//C example
//*****************************************************************************
// <propertieseditor>
//*****************************************************************************
// <o> Program Entry Point
PC = 0x4000000;

// <s> Change ID
// <s1.30> Change Password String
#define ID  "My User ID"
char pw[] = "My Password";
//*****************************************************************************
// </propertieseditor>
//*****************************************************************************
```

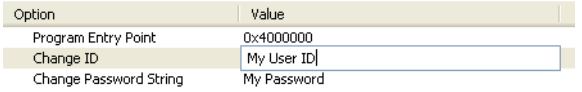

图 5-3 密码字符串 GUI 组件

数据读取协议示例

示例 5-3 中的示例代码生成如图 5-4 所示的 GUI 组件。

示例 5-3 数据读取协议控制标记

```
//C example
//*******************************************************************************
// <propertieseditor>
//*******************************************************************************
//   <q1.4>        DRP: Data Read Protocol
//                      <0=> Standard Read
//                      <1=> Early Read
DATAREAD(0xFFE00024, 0x10);   // EBI_MCR:  Data Read Protocol
//*******************************************************************************
// </propertieseditor>
//*******************************************************************************
```

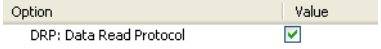

图 5-4 数据读取协议 GUI 组件

堆栈配置示例

示例 5-4 中的示例代码生成如图 5-5 所示的 GUI 组件。

示例 5-4 堆栈配置控制标记

```
; Assembler example
;*******************************************************************************
; <propertieseditor>
;*******************************************************************************
; <h> Stack Configuration (Stack Sizes in Bytes)
;   <o0> Undefined Mode      <0x0-0xFFFFFFF0:8>
;   <o1> Supervisor Mode     <0x0-0xFFFFFFF0:8>
;   <o2> Abort Mode          <0x0-0xFFFFFFF0:8>
;   <o3> Fast Interrupt Mode <0x0-0xFFFFFFF0:8>
;   <o4> Interrupt Mode      <0x0-0xFFFFFFF0:8>
;   <o5> User/System Mode    <0x0-0xFFFFFFF0:8>
; </h>

UND_Stack_Size  EQU     0xe0
SVC_Stack_Size  EQU     0x1000
ABT_Stack_Size  EQU     0xa8
FIQ_Stack_Size  EQU     0x40
IRQ_Stack_Size  EQU     0x68
USR_Stack_Size  EQU     0x3f8

Stack_Size      EQU     (UND_Stack_Size + SVC_Stack_Size + ABT_Stack_Size + \
                         IQ_Stack_Size + IRQ_Stack_Size + USR_Stack_Size)

                AREA    STACK, NOINIT, READWRITE, ALIGN=4
Stack_Mem       SPACE   Stack_Size

Stack_Top       EQU     Stack_Mem + Stack_Size
;*******************************************************************************
; </propertieseditor>
;*******************************************************************************
```

图 5-5 堆栈配置 GUI 组件

VPBDIV 定义示例

示例 5-5 中的示例代码生成如图 5-6 所示的 GUI 组件。

示例 5-5 VPBDIV 定义控制标记

```
; Assembler example
*******************************************************************************
; <propertieseditor>
;*******************************************************************************
; VPBDIV definitions
VPBDIV          EQU     0xE01FC100      ; VPBDIV Address

; <e> VPBDIV Setup
; <i> Peripheral Bus Clock Rate
;
;    NOTE: Option tag <e> uses index 0 as default if not specified
;          Enables or disables group of options using setup variable
;
;     <o1.0..1>   VPBDIV: VPB Clock
;                 <0=> VPB Clock = CPU Clock / 4
;                 <1=> VPB Clock = CPU Clock
;                 <2=> VPB Clock = CPU Clock / 2
;     <o1.4..5>   XCLKDIV: XCLK Pin
;                 <0=> XCLK Pin = CPU Clock / 4
;                 <1=> XCLK Pin = CPU Clock
;                 <2=> XCLK Pin = CPU Clock / 2
; </e>
VPBDIV_SETUP    EQU     0
VPBDIV_Val      EQU     0x20
;*******************************************************************************
; </propertieseditor>
;*******************************************************************************
```

图 5-6 显示一些因禁用了主组标题而显示为灰色的 GUI 组件。 若要启用该组，可以单击 VPBDIV Setup 的 GUI 框，或编辑相关代码，如下所示：

```
VPBDIV_SETUP    EQU     1
```

图 5-6 VPBDIV 定义 GUI 组件

5.4 分散文件编辑器

使用分散文件编辑器可以方便地创建和编辑分散加载描述文件，以供 ARM 链接器用于构造映像的内存映射。 该编辑器提供文本编辑器、分层树和映像的区和输出节的图形视图。 在“Preferences/首选项”对话框中可以更改缺省语法格式设置和颜色方案。 有关详细信息，请参阅第2-16 页的*“Preferences/首选项”对话框*。

如果它不是缺省编辑器，请在“Project Explorer/项目资源管理器”视图中右击源文件，然后从上下文菜单中选择 **Open With/打开方式 → Scatter File Editor/分散文件编辑器**。

分散文件编辑器显示以下选项卡：

Source/源代码 源代码的文本视图，具有语法突出显示和格式设置功能。

Regions/Sections/区/节

显示加载和执行内存映射的图形视图。 这些是不可编辑的，不过，您可以选择加载区来显示执行区中的相关内存块。

分散文件编辑器还通过“Outline/大纲”视图提供一个分层树，其中包含关联工具栏和上下文菜单。 在“Outline/大纲”视图中单击某个区或节可将编辑器的焦点移至代码中的相关位置。 如果此视图不可见，请从“Window/窗口”菜单中选择 **Show View/显示视图 → Outline/大纲**。

注意

有关如何使用分散加载描述文件的详细信息，请参阅《RealView® 编译工具链接器用户指南》和《RealView 编译工具链接器参考指南》。

分散文件编辑器不支持预处理指令，例如：

```
#! armcc -E.
```

在使用分散加载描述文件之前，必须在“Properties/属性”对话框的 **C/C++ Build/C/C++ 生成 → Settings/设置 → Tool settings/工具设置 → ARM RealView Linker/ARM RealView 链接器 → Output/输出**面板中将 `--scatter=file` 选项添加到项目。 有关详细信息，请参阅第2-17 页的*属性对话框*。

5.4.1 分散加载描述文件的示例

示例 5-6 是一个简单分散加载描述文件的示例，第5-18 页的示例 5-7 则是一个复杂分散加载描述文件的示例。

1. 使用现有项目或创建新项目。 有关详细信息，请参阅第 3 章 *使用项目*。
2. 创建一个新项目，并添加一个新的扩展名为 .scat 的空文本文件。例如 scatter.scat。 有关详细信息，请参阅第3-5 页的*创建新 RealView 项目*和第3-15 页的*向项目中添加文件*。
3. 使用“Outline/大纲”视图，单击工具栏图标，或右击并从上下文菜单中选择 **Add load region/添加加载区**。
4. 图 5-7 显示 **Add load region/添加加载区**对话框。 输入加载区名称，例如 LR1。 单击 **OK/确定**。

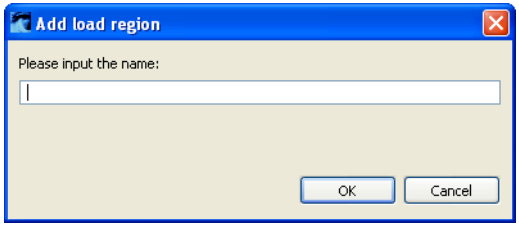

图 5-7 添加加载区名称

示例 5-6 简单的分散加载描述文件

```
LR1 0x0000 0x8000
{
    LR1_er1 0x0000 0x8000
    {
        * (+RO)
    }

    LR1_er2 0x10000 0x6000
    {
        * (+RW,+ZI)
    }
}
```

示例 5-7 复杂的分散加载描述文件

```
LR1 0x0
{
    LR1_er1 0x0
    {
        program.o (+RO)
    }

    LR1_er2 0x18000 0x8000
    {
        program1.o (+RW,+ZI)
    }
}

LR2 0x4000
{
    LR2_er1 0x4000
    {
        program2.o (+RO)
    }

    LR2_er2 0x8000 0x8000
    {
        program2.0 (+RW,+ZI)
    }
}
```

5.5 ELF 内容编辑器

ELF 内容编辑器通过特定 `fromelf` 命令行选项生成相关数据，为所选 ELF 文件创建窗体和图形视图。 使用该编辑器可以查看映像文件、目标文件和库文件的内容。 该编辑器是只读的，无法用于修改任何文件的内容。

如果它不是缺省编辑器，请在“Project Explorer/项目资源管理器”视图中右击源文件，然后从上下文菜单中选择 **Open With/打开方式 → ELF Content Editor/ELF 内容编辑器**。

ELF 内容编辑器显示以下选项卡中的一个或多个选项卡，具体取决于所选的文件类型：

Overview/概览

映像或目标文件显示头文件信息和节详细信息。

库文件显示所选库文件中每个对象的数据类型细分。

Symbol Table/符号表

显示所有符号的细分情况的表格视图。 数据可以保存到逗号分隔值 (CSV) 文件中。

此选项卡只用于映像或目标文件。

Instruction Sizes/指令大小

包含指令用法和函数类型的图形视图。 数据可以保存到 CSV 文件中。

此选项卡只用于映像或目标文件。

Disassembly/反汇编

具有语法突出显示功能的文本视图。

此选项卡只用于映像或目标文件。

5.5.1 Overview/概览

Overview/概览选项卡根据所选文件类型显示不同的信息。 映像或目标文件显示输入文件中每一节的头文件信息和节详细信息。 库文件显示所选库文件中每个对象的数据类型细分。

映像或目标文件

对于映像或目标文件，**Overview/概览**选项卡使用 `fromelf --text -v` 命令行选项提供 ELF 头信息和节信息的窗体视图。

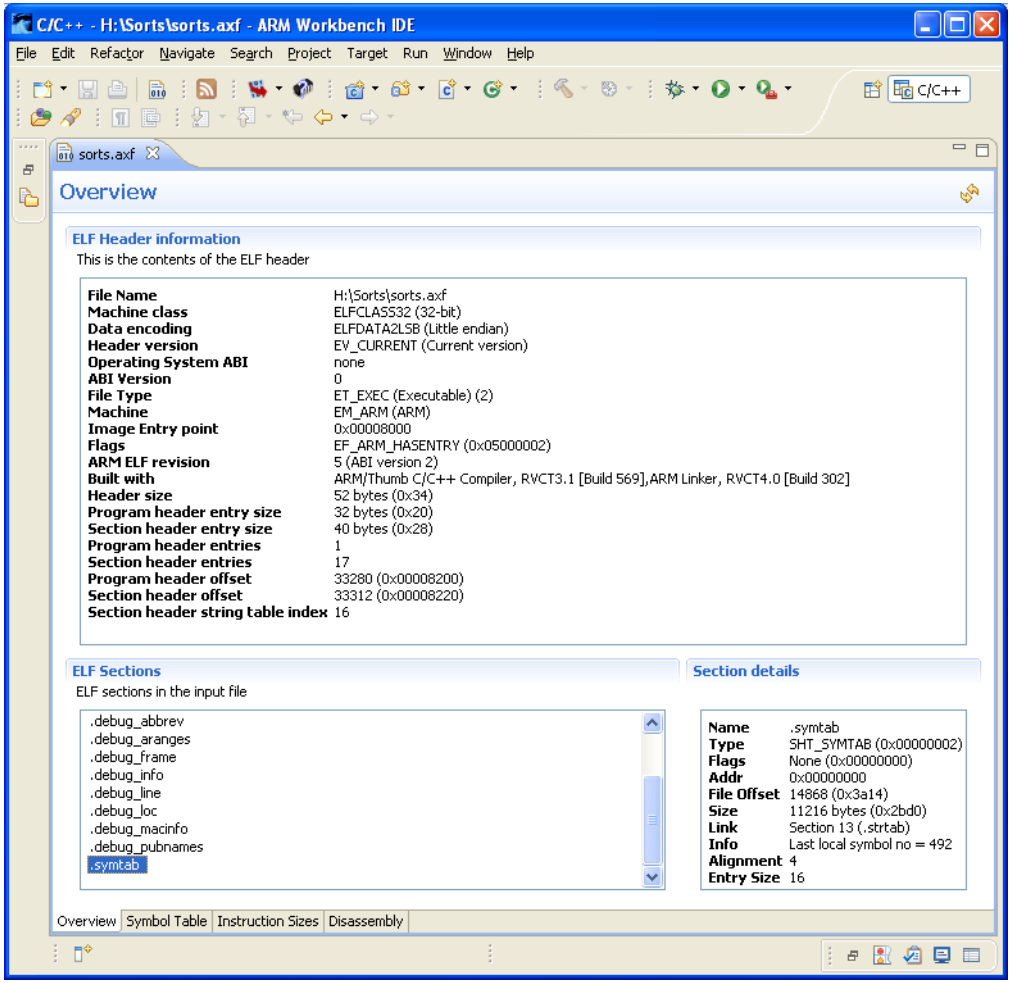

图 5-8 “Overview/概览”选项卡

库文件

对于库文件，**Overview/概览**选项卡使用 `armar --sizes` 命令行选项提供所选库文件中每个对象的细分。 数据可以显示为条形图或饼状图，也可以保存到 CSV 文件中。

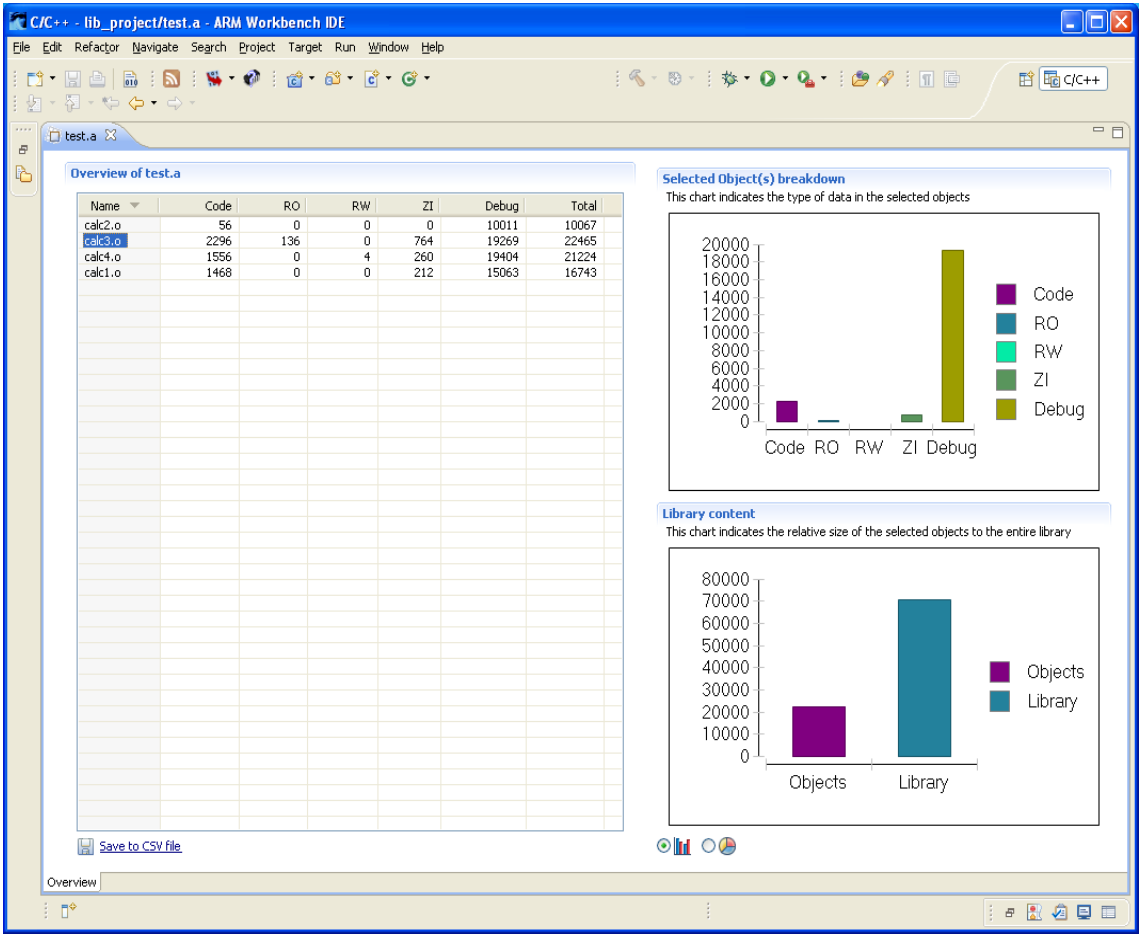

图 5-9 “Overview/概览”选项卡

5.5.2 Symbol table/符号表

对于映像或目标文件，**Symbol Table/符号表**选项卡使用 `fromelf --text -s` 命令行选项提供符号的表格视图。 可以按列对数据进行排序，也可以使用筛选选项隐藏映射符号。 数据可以保存到 CSV 文件中。

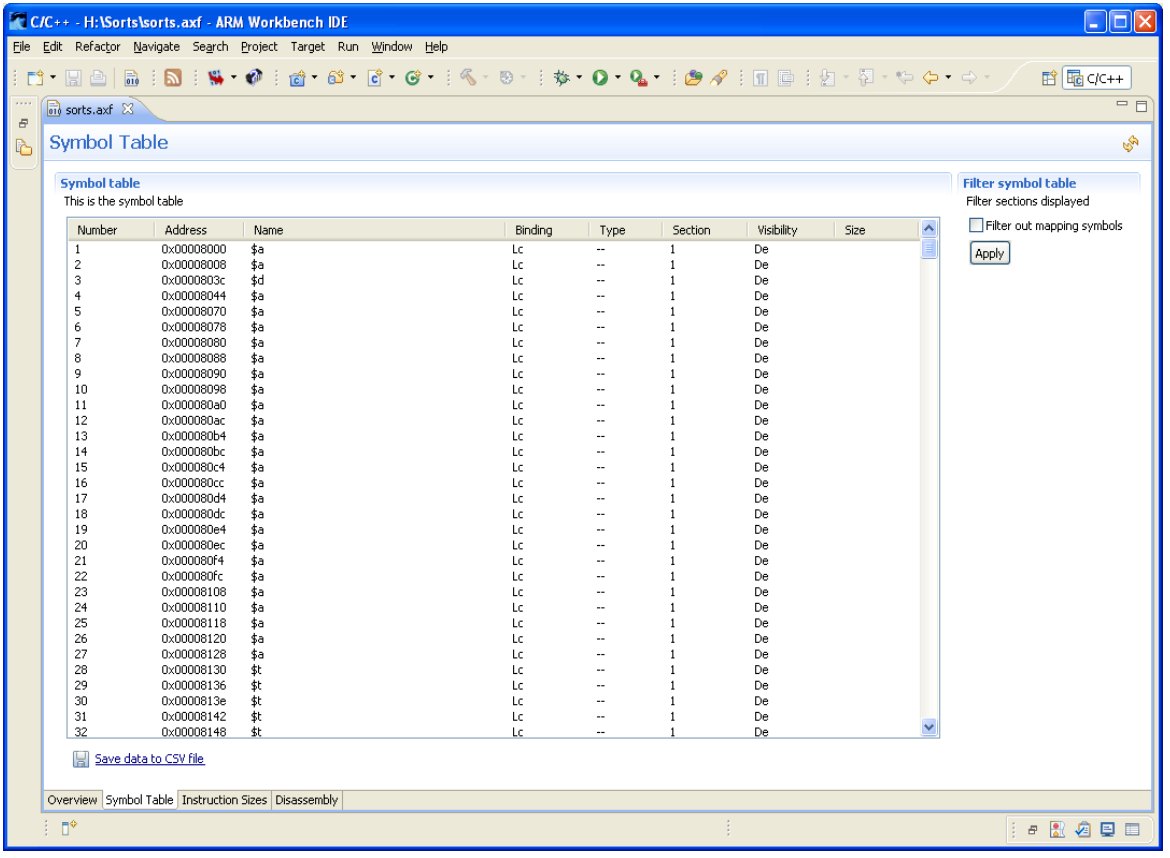

图 5-10 “Symbol Table/符号表”选项卡

5.5.3 Instruction sizes/指令大小

对于映像或目标文件，**Instruction Sizes/指令大小**选项卡使用 `fromelf --text --info=instruction_usage` 命令行选项提供所有指令的分类视图。

数据可以显示为条形图、饼状图或表格格式。 数据还可以保存到 CSV 文件中。

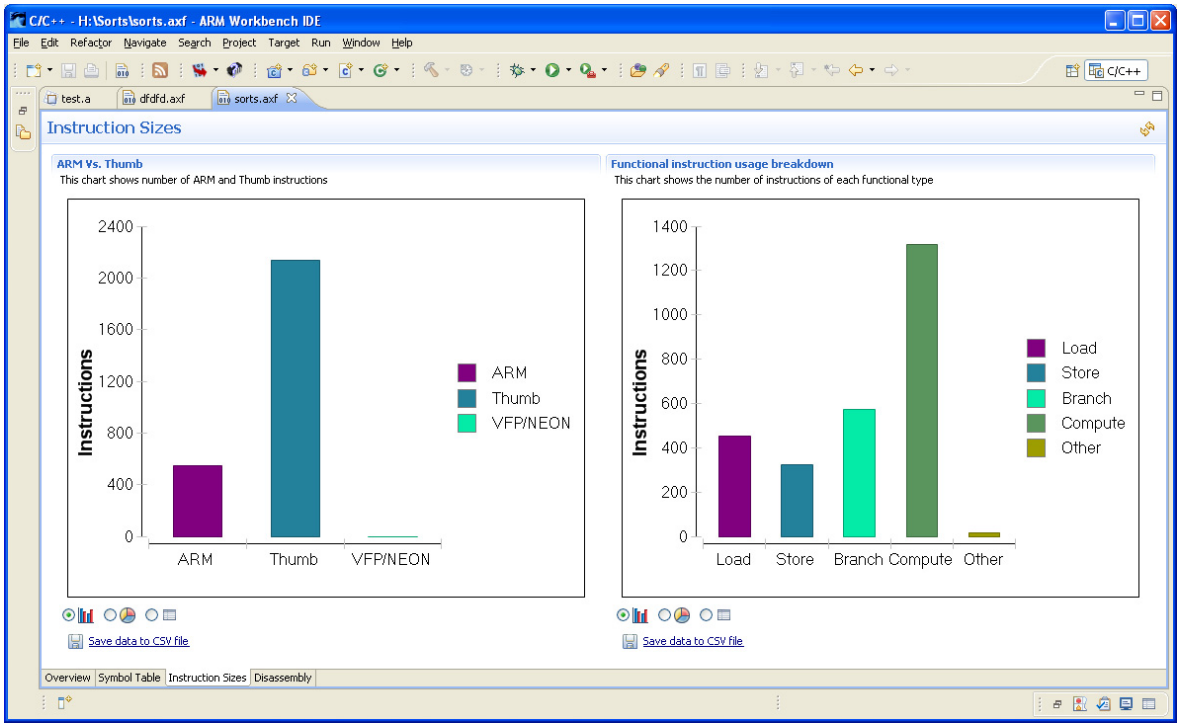

图 5-11 “Instruction Sizes/指令大小”选项卡

5.5.4 Disassembly/反汇编

对于映像或目标文件，**Disassembly/反汇编**选项卡使用 `fromelf --text -c` 命令行选项以语法突出显示的方式显示输出。 颜色方案和语法首选项使用与 ARM 汇编器编辑器相同的设置。 可以使用以下这些组合键在输出中导航：

- 使用 Ctrl+F 可打开“Find/查找”对话框以搜索输出
- 使用 Ctrl+Home 可将焦点移至输出的开始处
- 使用 Ctrl+End 可将焦点移至输出的结尾处

此外，还可以通过在“Disassembly/反汇编”视图中右击，在上下文菜单中选择 **Copy/复制**和 **Find/查找**选项。

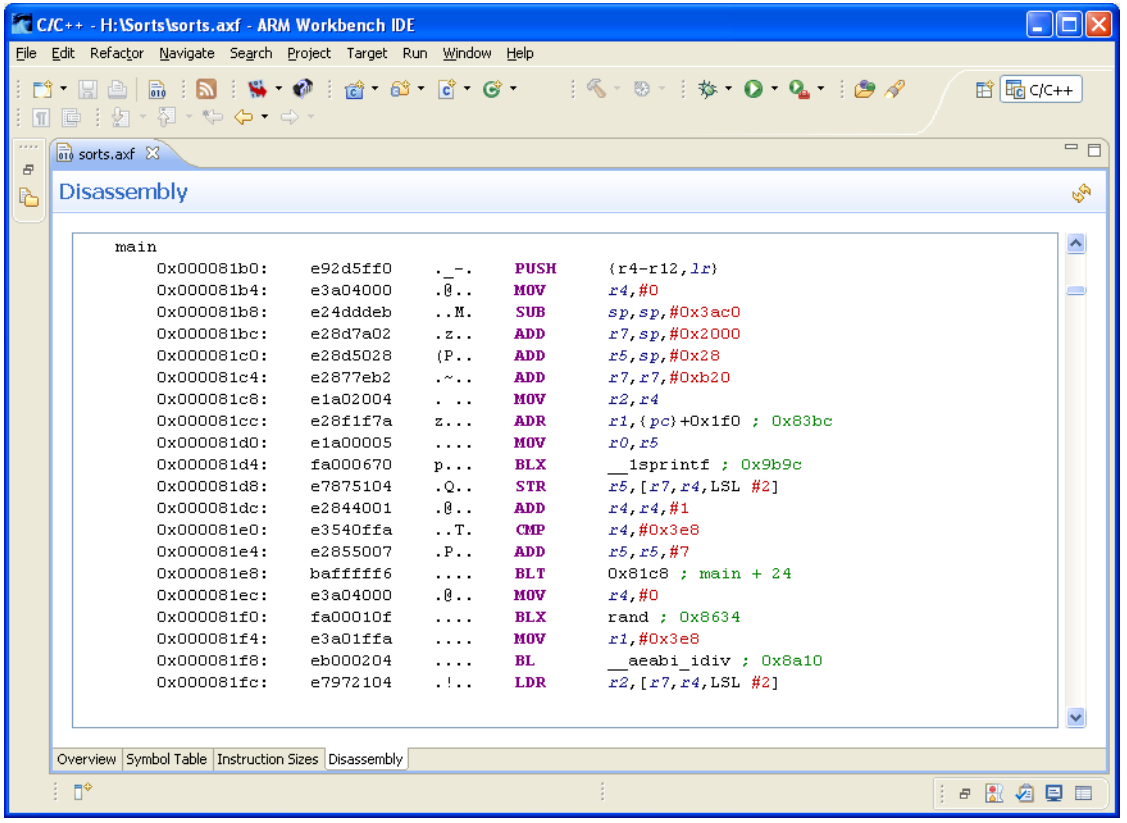

图 5-12 “Disassembly/反汇编”选项卡

第 6 章
使用 ARM Flash 编程器

本章介绍如何使用 ARM® Flash 编程器控制和更新所选目标上的闪存。

本章分为以下几节：

6.1 关于 ARM Flash 编程器

ARM Flash 编程器使用 RealView ICE 连接到目标，以便将 Flash 算法和 Flash 映像下载到 RAM（使用所需要的内存量）中。 您可以使用目标配置来指定 RAM 的基址和大小。 您可以修改 Flash 算法以适应特定 Flash 设备，也可以创建自己的 Flash 算法。 在执行算法时，映像从 RAM 复制到 Flash 设备。

若要将映像发送到 Flash 设备，必须设置以下内容：

- Flash 映像，表 6-1 列出了支持的映像类型
- Flash 编程器目标配置
- RealView ICE

表 6-1 将扩展名映射到映像类型

扩展名	映像类型
.bin	二进制映像
.axf、.elf	ELF 映像
.i32	Intel Hex-32
.m32	Motorola 32 位 S-record
.vhx	面向字节的（Verilog 内存模型）十六进制

ARM Flash 编程器的主要功能有：

- 使用“Send To/发送至”对话框可以将映像发送至 Flash 设备上的特定内存位置。 有关详细信息，请参阅第6-4 页的*对 Flash 设备进行编程*。
- 使用“Manage targets.../管理目标...”对话框可以设置与设备连接的 Flash 目标配置。 有关详细信息，请参阅第6-8 页的*管理 Flash 目标*。
- 使用“Flash Device Manager/Flash 设备管理器”对话框可以向工作区添加新 Flash 算法。 有关详细信息，请参阅第6-10 页的*使用 Flash 设备管理器*。
- 使用“New Flash Device Project/新建 Flash 设备项目”向导可以创建新 Flash 算法，以用于 ARM 项目或是导出和发布到第三方。 有关详细信息，请参阅第6-13 页的*创建新 Flash 算法*。

第6-3 页的图 6-1 中的流程图演示了这些功能及其互连方式。

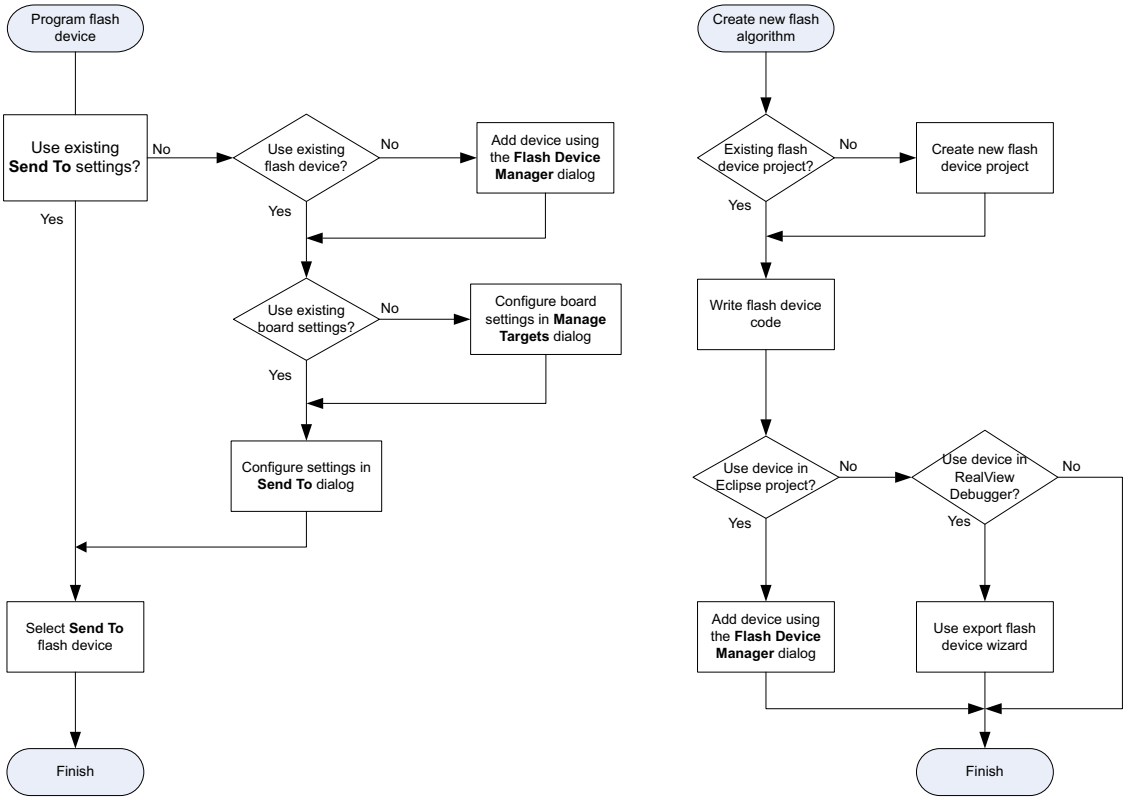

图 6-1 ARM Flash 编程器

6.2 对 Flash 设备进行编程

若要对设备上的闪存进行编程，请选择以下选项之一：

- 如果为设备设置过 Flash 编程器配置，则从“Run/运行”菜单选择 **Send To/发送至 → Send To/发送至 → *Flash Image/Flash 映像***。
- 如果未设置过 Flash 编程器配置：
  1. 在“Run/运行”菜单中单击 **Send To/发送至 → Send To/发送至...**。
  2. 使用“Send To/发送至”对话框中的选项卡可完成此过程，请参阅第6-5 页的图 6-2。

     Image/映像

     用于选择要发送至 Flash 设备的映像。 Flash 映像必须在 ARM 项目或项目子文件夹中，才能在此选项卡中将其选中。

     Connection/连接

     用于设置 Flash 设备的连接方法。 如果已配置了一个目标，则选择该目标，否则从工具栏中选择 **Add new RealView ICE target/添加新 RealView ICE 目标**，然后单击 **Configure.../配置...**，设置连接属性。

     Flash device/Flash 设备

     用于设置 Flash 设备的目标详细信息。 选择目标配置，然后选择相关 Flash 设备（一个 NAND 或所有 NOR 设备）。

     Program/编程

     用于设置 Flash 设备的擦除方法和验证。

     单击 **Apply/应用**可保存当前配置，但不会连接到 Flash 设备。

     单击 **Revert/还原**可撤消所有更改，还原为上次保存的配置。

     单击 **Send To/发送至**可连接到 Flash 设备，并发送所选 Flash 映像。

     有关每个选项卡中可用选项的详细信息，请使用动态帮助。 有关详细信息，请参阅第2-24 页的*动态帮助*。

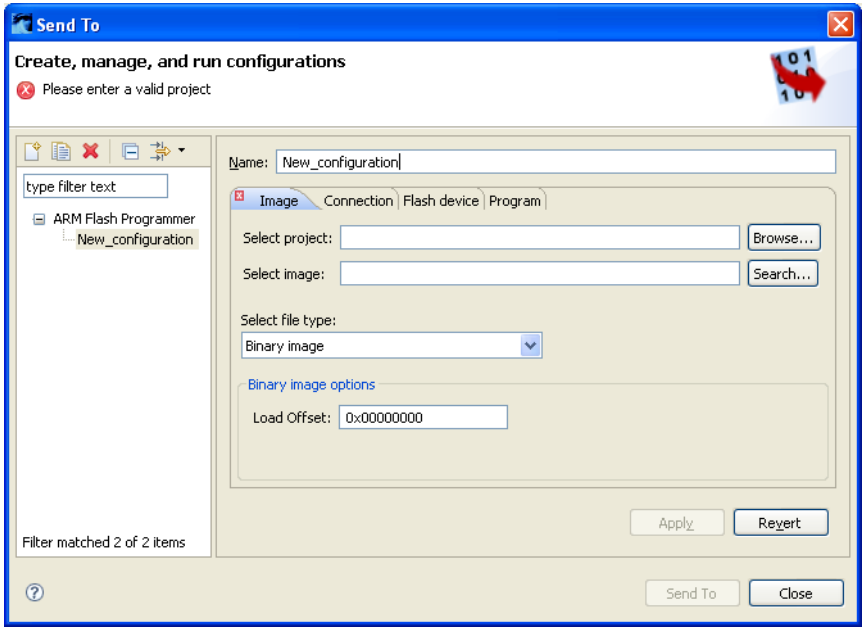

图 6-2 创建、管理和运行配置

3. 单击 **Send To/发送至**可将 Flash 映像发送至 Flash 设备。

注意

对 Flash 设备进行编程可能需要较长时间！

每个编程操作完成时，“Program Devices/对设备进行编程”对话框和主“Console/控制台”视图会进行更新。 完成后，结果显示在“Flash Programmer Results/Flash 编程器结果”对话框中，如图 6-3 所示。

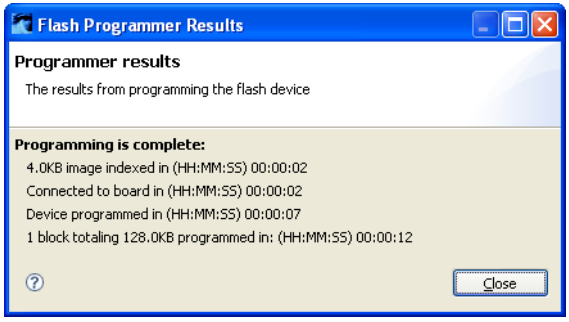

图 6-3 对设备进行编程的结果

6.3 导入 Flash 映像

Flash 映像必须在 ARM 项目或项目子文件夹中，才能使用 ARM Flash 编程器。

本节介绍如何将 Flash 映像从外部文件夹导入到当前工作区的现有项目中。

1. 从 **File/文件**菜单中选择 **Import.../导入...**，然后选择 **Flash Programmer/Flash 编程器** → **Flash Image/Flash 映像**，请参阅图 6-4。 单击 **Next/下一步**。

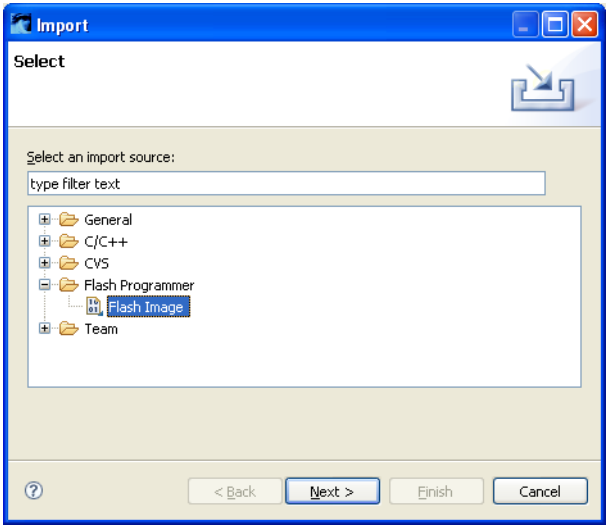

图 6-4 选择要导入的 Flash 映像

2. 选择映像源、目标项目以及相关导入选项。 单击 **Finish/完成**。

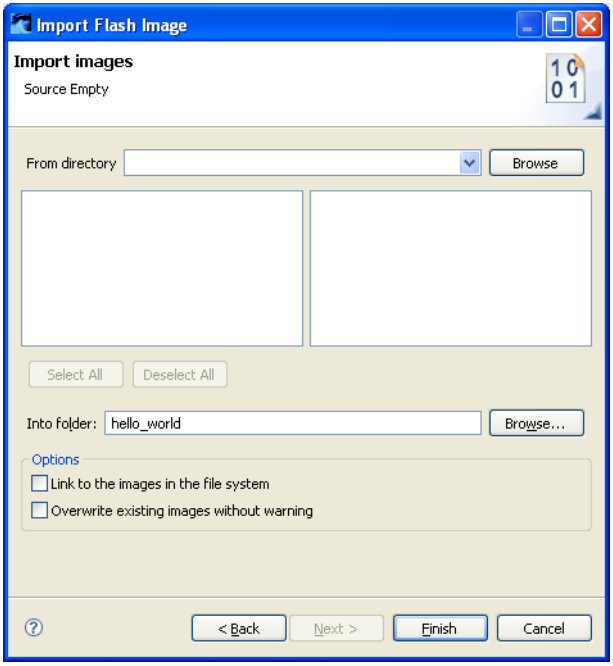

图 6-5 Flash 映像导入设置

6.4 管理 Flash 目标

若要从 Workbench 访问 Flash 设备，必须设置 Flash 目标配置。 一个 Flash 目标配置可以有多个关联的 Flash 设备。 ARM Flash 编程器插件提供了几个内置配置，还可以创建更多配置。

若要编辑或查看 Flash 目标配置，请从 **Target/目标**菜单中选择 **Manage Targets.../管理目标...**。图 6-6 显示了“Manage Targets/管理目标”对话框。

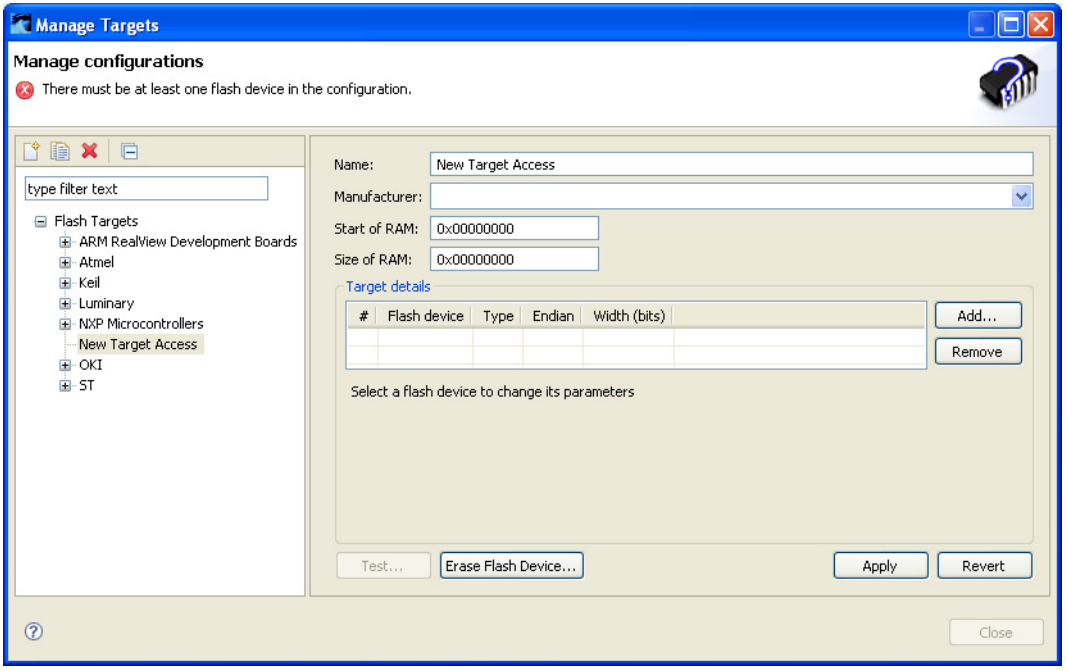

图 6-6 管理配置

1. 从树中选择 Flash 目标配置。 若要创建新配置，请单击工具栏或右击现有配置，然后选择 **New/新建**或 **Duplicate/复制**。

2. 设置完成此过程所需的目标详细信息表和 Flash 设备参数。 有关每个配置选项的信息，请访问动态帮助。 有关详细信息，请参阅第2-24 页的*动态帮助*。

   单击 **Apply/应用**可保存当前配置。 这不会连接到 Flash 设备。

   单击 **Revert/还原**可撤消所有更改，还原为上次保存的配置。

3. 若要测试设备，请单击 **Test.../测试...**，访问“Test Target/测试目标”对话框，如图 6-7 所示。

   a. 单击 **Configure.../配置...**，设置 **Connection method/连接方法**和 **CPU** 设置（如果尚未设置）。

   b. 单击 **Test device.../测试设备...**，对所需块进行擦除、编程和验证。

   注意

   对 Flash 设备进行测试可能需要较长时间。

   在每个编程操作完成时，“Progress Information/进度信息”对话框和主“Console/控制台”视图会进行更新。 完成后，结果显示在“Flash Programmer Results/Flash 编程器结果”对话框中，如图 6-8 所示。

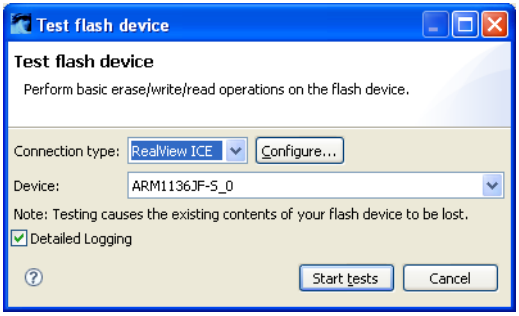

图 6-7 测试目标连接方法

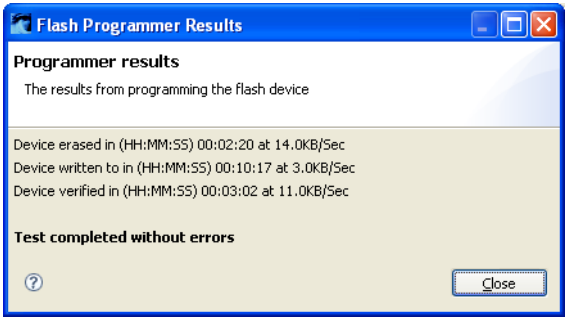

图 6-8 设备测试结果

6.5 使用 Flash 设备管理器

新 Flash 算法必须在当前工作区中，才能从“Manage Targets/管理目标”对话框访问该算法，或将其导出供 RealView Debugger 使用。

本节介绍如何向工作区添加新 Flash 算法。

1. 从 **Target/目标**菜单中选择 **Flash Device Manager/Flash 设备管理器**，请参阅图 6-9。

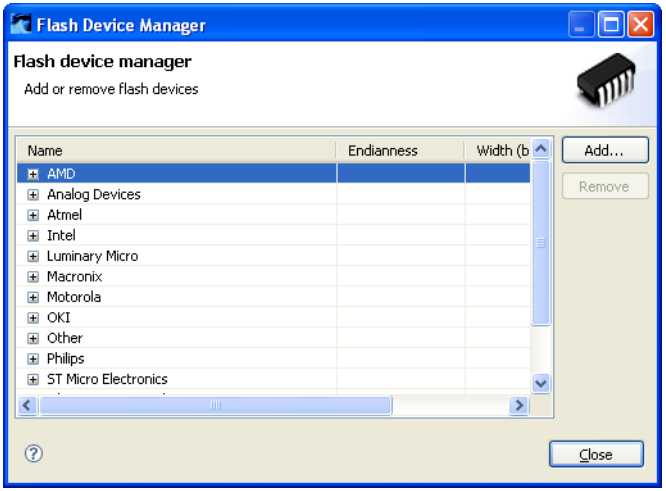

图 6-9 添加或删除 Flash 设备

2. 单击 **Add.../添加...**，打开“Import Flash Device/导入 Flash 设备”对话框，向工作区添加新 Flash 算法。

   注意

   若要删除 Flash 算法，请选择算法的名称，然后单击 **Remove/删除**。 不能删除内置 Flash 算法。

3. 选择包含 Flash 算法的项目，请参阅第6-11 页的图 6-10。

   单击 **Next/下一步**。

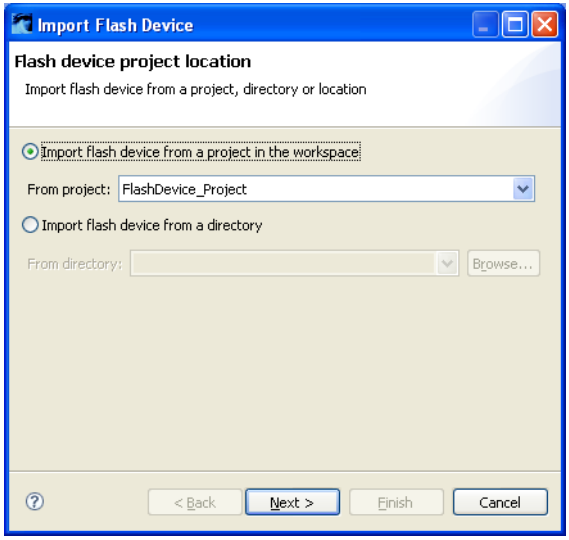

图 6-10 导入 Flash 设备

4. 选择生成配置，以及与 Flash 设备关联的相关目标文件。 建议选择完全优化的生成（如发布配置），这样，Flash 操作的速度会更快。 单击 **Finish/完成**。

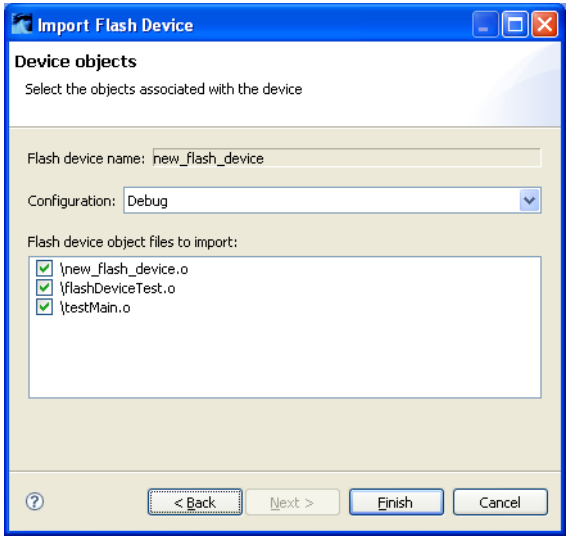

图 6-11 导入 Flash 设备对象文件

5. 现在，Flash 算法显示在“Flash Device Manager/Flash 设备管理器”对话框中，可由 ARM Flash 编程器使用。 单击 **Close/关闭**。

6.6 创建新 Flash 算法

Flash 设备项目可用于创建新 Flash 算法，以发布到第三方或导入其他 ARM 项目中。 Flash 设备项目向导创建新项目文件夹，其中包含几个 .h 和 .c 文件。

本节介绍如何创建新 Flash 算法：

1. 从“File/文件”菜单中选择 **New/新建 → Project.../项目...**。
2. 选择 **Flash Programmer/Flash 编程器 → New RealView Flash Device Project/新建 RealView Flash 设备项目**，请参阅图 6-12。 单击 **Next/下一步**。

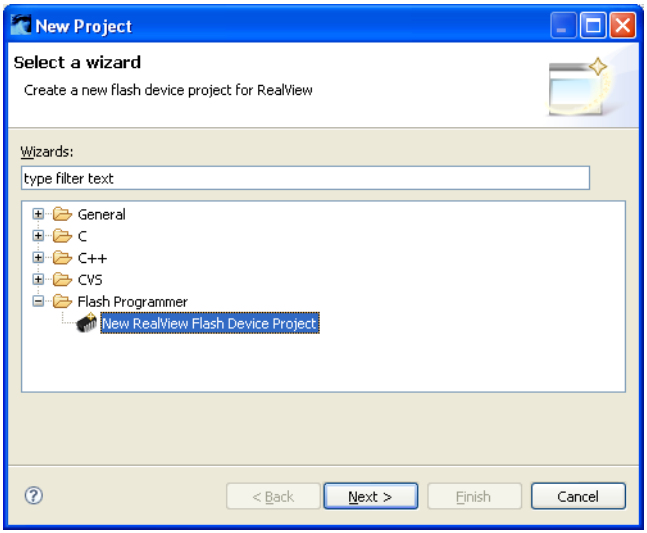

图 6-12 新建 Flash 设备项目

3. 输入 Flash 设备项目的名称，请参阅第6-14 页的图 6-13。

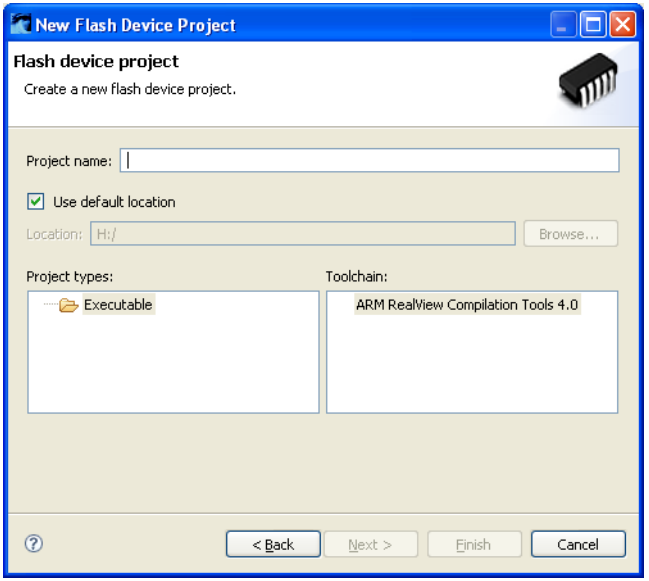

图 6-13 为 Flash 项目命名

4. 使 **Use default location/使用缺省位置**选项保持为选中状态，以便在显示的缺省文件夹中创建项目。 或者，取消选择此选项，然后浏览至所需的项目文件夹。 单击 **Next/下一步**。

5. 根据需要为项目输入 Flash 设备详细信息，请参阅第6-15 页的图 6-14。 请确保在“Flash Device Name/Flash 设备名称”字段中输入有意义的名称。 单击 **Next/下一步**。

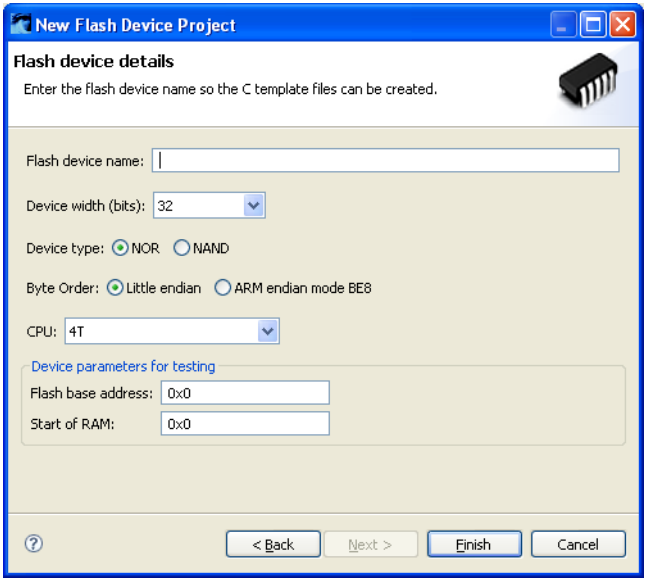

图 6-14 Flash 设备详细信息

注意

通过“Properties/属性”对话框中的“Flash Device/Flash 设备”选择，可以在以后修改这些 Flash 设备详细信息。

6. 在“Configurations/配置”面板中，使 **Debug/调试**和 **Release/发布**选项保持为选中状态，以便为调试和发布配置保存不同项目设置，请参阅第6-16 页的图 6-15。 单击 **Finish/完成**。

注意

如果选择了 **Build Automatically/自动生成**，则在单击 **Finish/完成**时会自动生成项目。 如果未选择该选项，则必须从 **Project/项目**菜单中选择 **Build Project/生成项目**才能完成生成步骤。

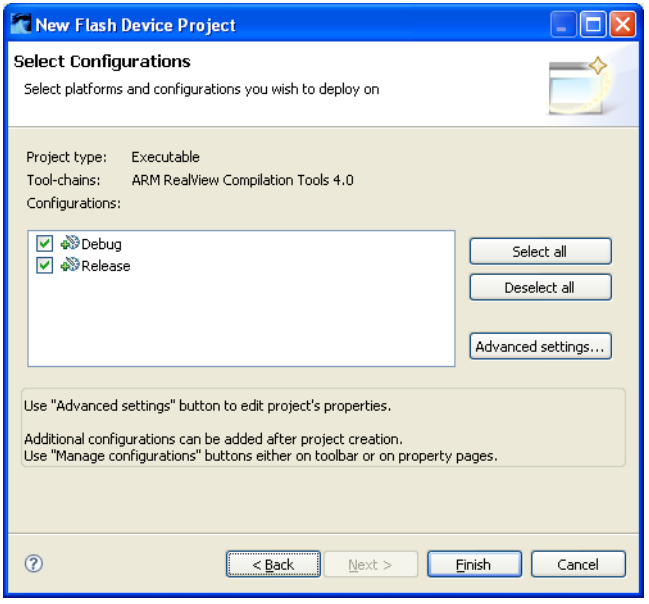

图 6-15 Flash 设备配置设置

新项目会显示在“Project Explorer/项目资源管理器”视图中。 Flash 设备项目向导创建以下文件并将这些文件放置在项目文件夹中：

XXXXX.c

使用为 Flash 设备输入的名称创建的模板 .c 文件。 必须编辑此文件才能实现 Flash 算法。

有关详细信息，请参阅《应用程序说明 190：使用 Eclipse 创建 Flash 算法》(*Application Note 190 Creating Flash Algorithms with Eclipse*)。

flashDeviceTest.h

声明测试工具的头文件。 不要修改此文件。

flashDeviceTest.c

测试工具的实现。 可以在调试器中单步调试此代码，从而调试 Flash 算法。 可以编辑此文件中的一些选项以更改测试的行为。

`testMain.c`

定义在链接 Flash 算法的“Debug/调试”配置时使用的 `main()` 函数。 `main()` 函数调用测试工具，同时传入 Flash 设备的地址。 不要修改此文件。

`dummyMain.c`

定义在链接 Flash 算法的“Release/发布”配置时使用的 `main()` 函数。 不要修改此文件。

6.7 导出供 RealView Debugger 使用的板

可以导出使用 ARM Flash 编程器创建的新 Flash 设备板，以供 RealView Debugger 使用。 本节介绍如何创建所需的板/芯片定义 (BCD) 文件。

1. 从 **File/文件**菜单中选择 **Export.../导出...**，然后选择 **Flash Programmer/Flash 编程器 → RealView Debugger Board/RealView Debugger 板**，请参阅图 6-16。 单击 **Next/下一步**。

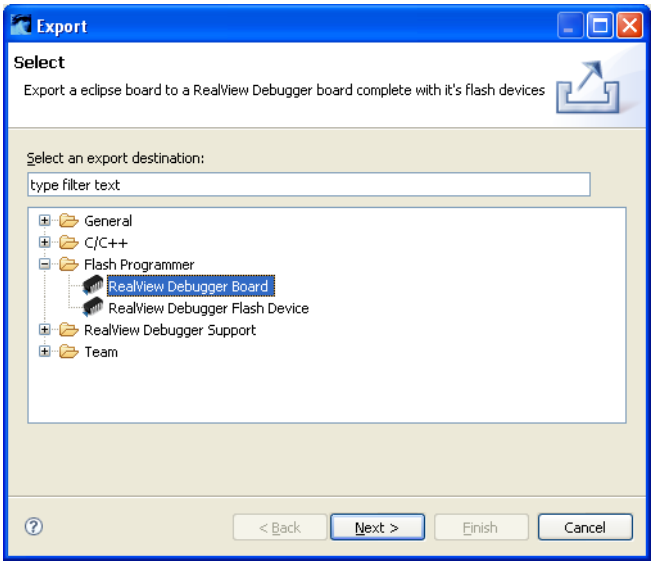

图 6-16 导出 RealView Debugger 板

2. 选择所需的板和设备，请参阅第6-19 页的图 6-17。 单击 **Finish/完成**将所选的板和设备导出到 RealView Debugger。

   生成的 BCD 文件会导出到 RealView Debugger 缺省文件夹：
   install_directory\RVD\Core\...\etc

   启动 RealView Debugger 后，可以在“Connection Properties/连接属性”窗口中选择新板。 有关详细信息，请参阅《RealView Debugger 目标配置指南》(*RealView Debugger Target Configuration Guide*)。

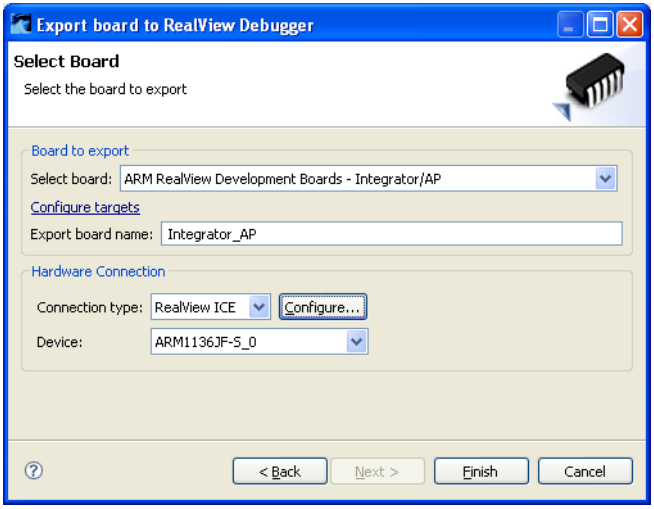

图 6-17 选择板配置

6.8 导出供 RealView Debugger 使用的 Flash 设备

可以导出使用 ARM Flash 编程器创建的新 Flash 算法，以供 RealView Debugger 使用。 本节介绍如何创建所需的 Flash 方法 (FME) 文件。

1. 从 **File/文件**菜单中选择 **Export.../导出...**，然后选择 **Flash Programmer/Flash 编程器 → RealView Debugger Flash Device/RealView Debugger Flash 设备**，请参阅图 6-18。 单击 **Next/下一步**。

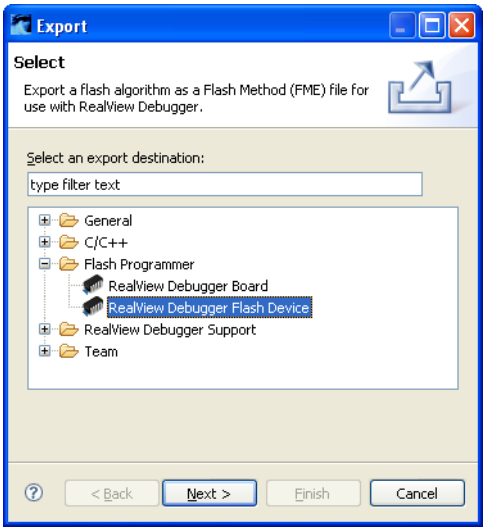

图 6-18 将 Flash 设备导出到 RealView Debugger

2. 选择要导出的 Flash 设备文件。 这些文件可以来自已安装的设备或项目。 输入目标 FME 文件的名称和位置，或使用 **Browse.../浏览...** 找到文件，请参阅第6-21 页的图 6-19。 单击 **Next/下一步**。

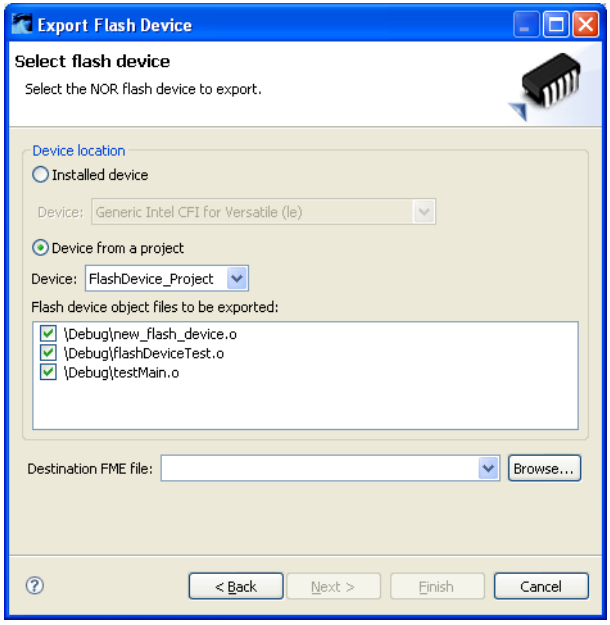

图 6-19 导出 Flash 设备目标文件

3. 通过以下任一方式为 Flash 设备输入块结构：
   - 使用 **Add.../添加...** 和 **Edit.../编辑...** 按钮为 Flash 设备输入块结构。图 6-20 和第6-22 页的图 6-21 显示设备结构和块详细信息。 完成设备结构后，单击 **Next/下一步**。

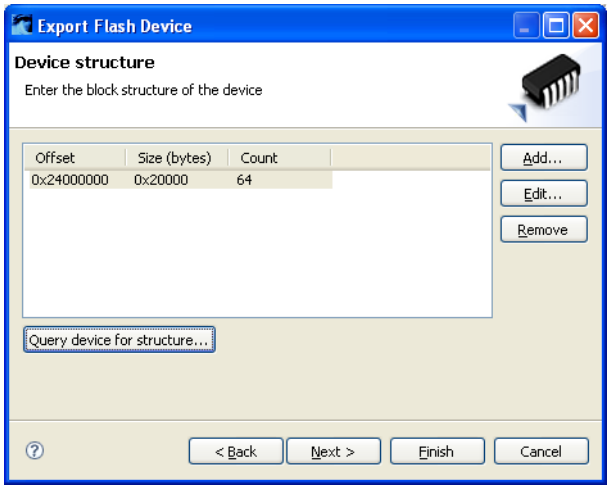

图 6-20 导出 Flash 设备结构

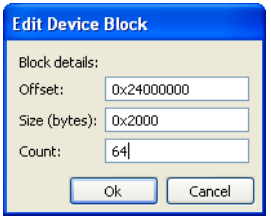

图 6-21 导出 Flash 设备块

- 对于已安装的设备，可以单击 **Query device for structure.../查询设备的结构...** 自动为 Flash 设备设置块结构。

  配置目标连接，单击 **Query device.../查询设备...** 自动从 Flash 算法读取块结构，请参阅图 6-22。 完成设备结构后，单击 **Next/下一步**。

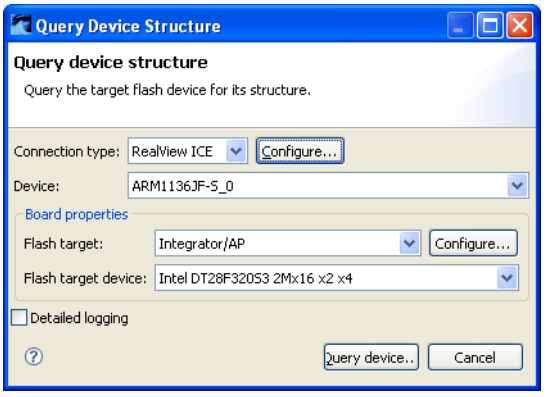

图 6-22 查询设备结构

4. 第6-23 页的图 6-23 显示的是参数对话框。 输入设备参数，然后单击 **Finish/完成**保存 FME 文件。

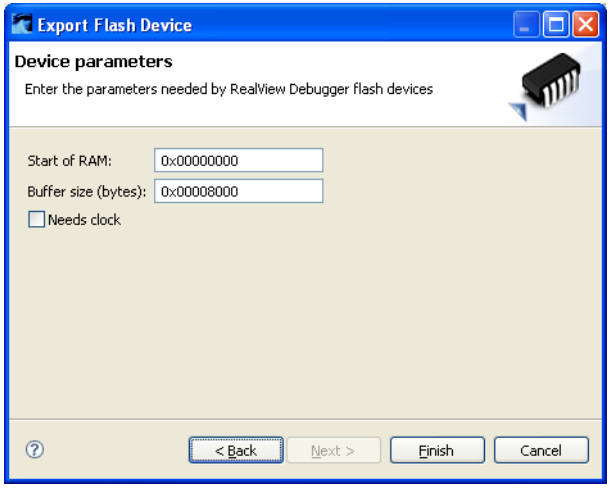

图 6-23 导出 Flash 设备参数

注意

必须创建新的板/芯片定义 (BCD) 文件才能引用新 Flash 设备的导出 FME 文件。有关详细信息，请参阅《RealView Debugger 目标配置指南》(*RealView Debugger Target Configuration Guide*)。

第 7 章
使用 RealView Debugger

本章介绍如何在 Workbench 中启动 RealView Debugger，及如何配置目标连接设置以传输到该调试器。

本章分为以下几节：

7.1 将可执行映像加载到 RealView Debugger 中

利用 Workbench 可以设置目标配置并将其与项目文件一起进行保存。 这些配置文件存储在项目中的 rvd 子文件夹中。

如果项目包含目标配置文件，则会自动向 RealView Debugger 传送连接详细信息。 有关详细信息，请参阅*使用现有的 Workbench 目标配置进行加载*。

如果项目不包含目标配置文件，则必须手动设置与目标的连接。 有关详细信息，请参阅第7-3 页的*在无 Workbench 目标配置的情况下加载*。

7.1.1 使用现有的 Workbench 目标配置进行加载

在从 Workbench 中启动 RealView Debugger 以运行或调试可执行映像之前，应确保 RealView Debugger 处于关闭状态。 在从 Workbench 中启动 RealView Debugger 时，将自动配置 RealView Debugger 中的连接属性并将所有控制传递给 RealView Debugger。 您必须使用 RealView Debugger界面来执行步进、插入断点以及检查内存等调试操作。 有关使用 RealView Debugger 的信息，请参阅《RealView Debugger 用户指南》(*RealView Debugger User Guide*)。 将可执行映像加载到 RealView Debugger 中：

1. 在 Workbench 中，确保项目已生成并包含可执行映像（请参阅第2-19 页的*生成*）。
2. 在项目资源管理器视图中单击您的可执行映像文件。 从“Run/运行”菜单中选择 **Debug As/调试方式 → Load into RealView Debugger/加载到 RealView Debugger**。
3. 对于模拟目标，Workbench 将自动启动 RealView Debugger 并将可执行映像加载到其中。 如果目标不是模拟目标，则会自动配置 RealView Debugger 中的连接属性，但需要手动进行连接和加载。

Workbench 会记住上次加载的可执行映像。 如果需要从同一项目中重新加载该可执行映像，只需按键盘上的 F11 即可。 这会根据需要重新生成项目，并将可执行映像重新加载到 RealView Debugger 中。

——注意——

如果要生成并调试新的目标，则必须关闭 RealView Debugger，然后再加载其他可执行映像。 这将允许从 Workbench 中传送新的目标配置。

7.1.2 在无 Workbench 目标配置的情况下加载

可以从 Workbench 中启动 RealView Debugger 来运行或调试可执行映像。 首先必须在 RealView Debugger 中创建与目标的连接。 在从 Workbench 中启动 RealView Debugger 时，会将映像加载到目标中，同时将所有控制传递给 RealView Debugger。 您必须使用 RealView Debugger 界面来执行步进、插入断点以及检查内存等调试操作。 有关使用 RealView Debugger 的信息，请参阅 *RealView Debugger 用户指南*。 将可执行映像加载到 RealView Debugger 中：

1. 启动 RealView Debugger。
2. 在 RealView Debugger 中创建与目标的连接。
3. 关闭 RealView Debugger。
4. 在 Workbench 中，确保项目已生成并包含可执行映像（请参阅第2-19 页的*生成*）。
5. 在项目资源管理器视图中单击您的可执行映像文件。 从“Run/运行”菜单中选择 **Debug As/调试方式 → Load into RealView Debugger/加载到 RealView Debugger**。
6. Workbench 将自动启动 RealView Debugger 并将可执行映像加载到其中。

Workbench 会记住上次加载的可执行映像。 如果需要从同一项目中重新加载该可执行映像，只需按键盘上的 F11 即可。 这会根据需要重新生成项目，并将可执行映像重新加载到 RealView Debugger 中。

7.2 创建调试配置

您可以在 Workbench 中为每个可执行映像创建并设置自己的调试配置。 为可执行映像创建新的调试配置：

1. 在项目资源管理器视图中选择一个项目。
2. 从 **Run/运行**菜单中选择 **Open Debug Dialog.../打开调试对话框...**，以显示“Create, manage, and run configurations/创建、管理并运行配置”对话框。
3. 在左侧的配置面板中，选择 **RealView Debugger**。
4. 单击工具栏或右击并选择 **New/新建**，以便为项目创建新的调试配置。 可在“C/C++ Application/C/C++ 应用程序”字段中看到可执行映像的名称。

   注意

   如果项目中不包含可执行映像，或者包含多个可执行映像，则“C/C++ Application/C/C++ 应用程序”字段将保留为空白。 Workbench 将警告该程序不存在，并禁用面板上的 **Debug/调试**按钮。 请参阅第7-5 页的*选择另一个要调试的映像*，了解如何设置要调试的可执行映像。
5. 缺省情况下，将使用项目名称创建一个新的调试配置。 该调试配置将显示在“Configurations/配置”面板中的“RealView Debugger”下。

7.3 设置调试配置

要调试可执行映像，既可以创建新的调试配置，也可以使用现有的配置。 如果尚未创建调试配置，请参阅第7-4 页的*创建调试配置*，以创建新的配置。 本节介绍如何设置现有的调试配置。

7.3.1 选择现有的调试配置

选择现有的调试配置：

1. 从 **Run/运行**菜单中选择 **Open Debug Dialog.../打开调试对话框...**，以显示“Create, manage, and run configurations/创建、管理并运行配置”对话框。
2. 在“Configurations/配置”面板中，展开 RealView Debugger。
3. 选择要使用的调试配置。

Workbench 提供了多个面板，可用于设置或修改选定的 RealView Debugger 配置。 以下几节将介绍 **Main/主要**、**Arguments/参数**和 **Connection/连接**选项卡。

7.3.2 选择另一个要调试的映像

Main 选项卡提供的选项可将另一个项目或可执行映像与选定的调试配置关联在一起：

Project/项目 可以在“Project/项目”字段中键入项目名称，也可以单击 **Browse.../浏览...** 从可用项目的列表中进行选择。

——注意——

只能选择当前在 Workbench 中打开的项目。

C/C++ Application/C/C++ 应用程序

可以在“C/C++ Application/C/C++ 应用程序”字段中键入要调试的映像的名称，也可以单击 **Browse.../浏览...** 选择可执行映像。 使用 **Search Project.../搜索项目...** 可显示当前项目中可供选择的可执行映像的列表。 来自不同生成配置（例如调试和发布）的可执行映像将显示在此列表中。

7.3.3 配置 RealView Debugger 连接设置

使用 **Connection/连接**选项卡可以配置 RealView Debugger 加载选项：

Connection/连接

如果 RealView Debugger 已连接到多个目标，则可以使用 **Connections/连接**下拉列表选择映像要加载到的目标。 单击 **Get Connections/获取连接**，可使 Workbench 从 RealView Debugger 中获取可用连接的列表。 取消选择 **Load into first target/加载到第一个目标中**可查看可用的连接并选择所需的连接。 如果选中了 **Load into first target/加载到第一个目标中**，Workbench 会将可执行映像加载到可用连接列表中显示的第一个目标中。

Parts to load/要加载的部分

使用此选项可选择要将映像的哪些部分加载到目标中：

Symbols and Image/符号和映像

使用此选项可加载所有的调试符号和程序映像。

Image Only/仅映像

使用此选项可只加载程序映像，而不加载调试符号。

Symbols Only/仅符号

使用此选项可只加载符号，而不加载程序映像。

Loading mode/加载模式

使用此选项可选择是否替换目标中已有的可执行映像：

Append/追加 使用此选项可向现有映像附加新的可执行映像。

Replace/替换 使用此选项可将现有映像替换为正在加载的映像。

Sections 可以使用 Sections 字段指定在加载映像时要加载的节。 此字段通常用于在启动程序时重新加载已初始化的数据节。 选择 **Load all sections/加载所有节**作为缺省选项。

Set Program Counter (PC) to start address from object module/将程序计数器 (PC)设置为目标模块中的开始地址

可以使用此选项对 PC 进行设置，以使其成为 ELF 映像每次被加载到 RealView Debugger 中时指定的起始地址。

7.3.4 指定执行参数

在 **Arguments/参数**选项卡中可以指定可执行映像的参数：

Program arguments/程序参数

在“Program arguments/程序参数”字段中，可以为可执行映像指定一个以空格分隔的参数列表。

Variables/变量 可以将 Workbench 提供的变量用作可执行映像的参数。 单击 **Variables.../变量...** 可选择所需的变量，也可以创建自己的变量。

7.4 使用调试配置启动 RealView Debugger

对于要加载到 RealView Debugger 中的可执行映像，如果您不想更改其调试配置，请按照第7-2 页的*将可执行映像加载到 RealView Debugger 中*中的步骤操作。

在将可执行映像加载到 RealView Debugger 中之前，如果您希望更改任何调试配置设置，请按照下列步骤操作：

1. 确保 RealView Debugger 已连接至目标。 有关详细信息，请参阅 *RealView Debugger 用户指南*。 关闭仍在运行的所有 RealView Debugger 实例。
2. 从 **Run/运行**菜单中选择 **Debug/调试**。
3. 在“Configurations/配置”框中选择您的 RealView Debugger 配置，或创建新的调试配置。 请参阅第7-4 页的*创建调试配置*。
4. 如果您想更改要调试的项目或映像，请使用 **Main** 选项卡选择另一个项目或映像。
5. 如果要修改或传递可执行映像的参数，请使用 **Arguments/参数**选项卡。 请参阅第7-7 页的*指定执行参数*。
6. 选择 **Connections/连接**选项卡，然后单击 **Get Connections/获取连接**。 随即会启动 RealView Debugger，并且 Workbench 将尝试连接到 RealView Debugger。
7. 当 Workbench 连接至 RealView Debugger 后，会打开 Workbench“Debug/调试”对话框中的 **Connection/连接**选项卡。 如果已将多个目标连接至 RealView Debugger，请取消选择 **Load into first target/加载到第一个目标中**选项。 可用目标连接显示在 **Connection** 下拉列表中。 选择要使用的连接（请参阅第7-9 页的图 7-1）。 根据需要修改加载设置。 请参阅第7-6 页的*配置 RealView Debugger 连接设置*。

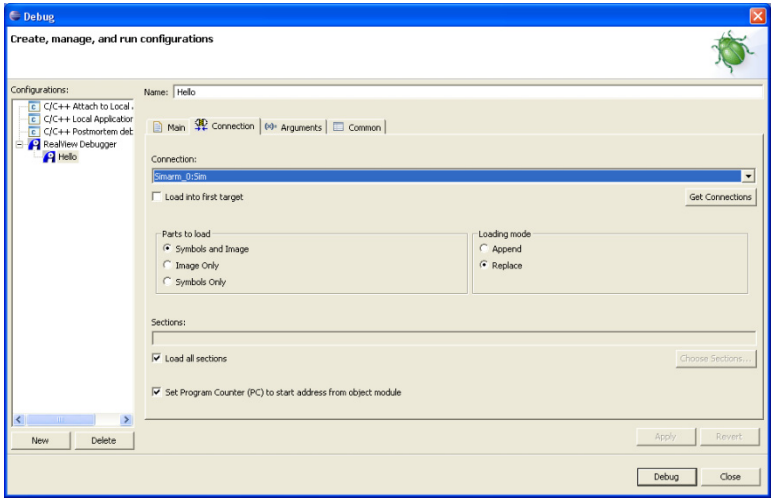

图 7-1 “Debug/调试”面板

8. 单击 **Debug/调试**。项目会根据需要重新生成。

9. 您的可执行映像将加载到目标中，随后便可以使用 RealView Debugger 对其进行调试了。

7.5 导出 IP-XACT 设计文件以供 RealView Debugger 使用

您可以导出用 IP-XACT 创建的设计文件以供 RealView Debugger 使用。 本节介绍如何创建所需的 BCD 文件。

1. 从 **File/文件**菜单中选择 **Export.../导出...**。
2. 选择 **RealView Debugger Support/RealView Debugger 支持 → IP-XACT to BCD/IP-XACT 到 BCD**，如图 7-2 所示。 单击 **Next/下一步**。

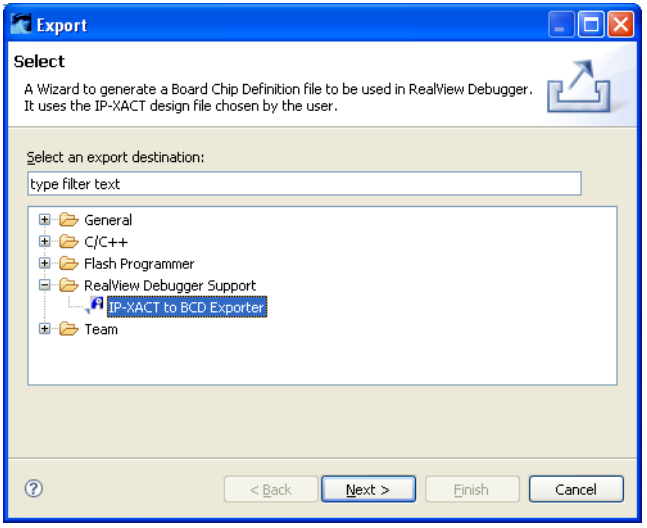

图 7-2 将 IP-XACT 导出到 RealView Debugger

3. 选择要导出的设计文件并输入目标 BCD 文件的位置，或使用 **Browse.../浏览...**，如第7-11 页的图 7-3 所示。 单击 **Finish/完成**保存该 BCD 文件。

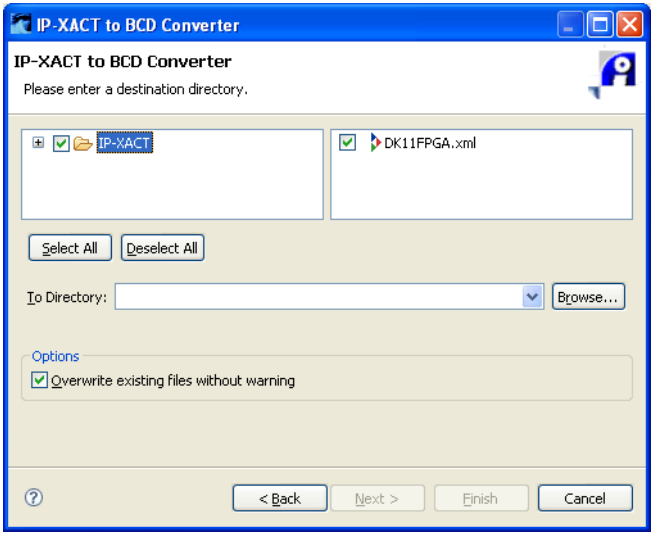

图 7-3 选择 IP-XACT 设计文件

附录 A
术语、快捷键和图标

本附录介绍《ARM Workbench IDE 用户指南》中使用的一些术语。 另外，还列出了在创建和生成 ARM® 项目时会用到的一些常见键盘快捷键以及菜单或工具栏图标。

本章分为以下几节：

A.1 术语

本节介绍《ARM Workbench IDE 用户指南》中使用的一些术语。 这些术语按字母顺序列出：

块 Flash 设备中可编程的细分单元。

对话框 一个小页面，其中包含选项卡、面板和可编辑字段，用于提示您输入信息。

编辑器 一个视图，用于控制特定文件类型的源代码的可视方面。

擦除 Flash 设备的一种功能，用于将存储单元重置为已知值。

Flash 设备 一组具有单个命令接口的闪存。

面板 对话框或选项卡中的一个小区域，用于对可编辑字段进行分组。

透视图 Workbench 窗口中的一个页面，其中包含一组相关视图、编辑器、菜单和工具栏。

编程 一个术语，用于描述将数据存储到 Flash 设备上的过程。

项目 Workbench 中的一组相关文件和文件夹。

资源 一个普通术语，用于描述项目、文件、文件夹或这些内容的组合。

发送至 一个术语，用于描述将文件发送至目标的过程。

选项卡 对话框中一个叠加的小页面，其中包含面板和可编辑字段，用于对相关信息进行分组。 单击某选项卡可将其显示在前端。

目标 文件要发送到的位置，例如一个 Flash 映像。

视图 一个小页面，用于显示特定功能的相关信息。

宽度 Flash 设备可在本机存取的最小位数（8、16 或 32）。

向导 一组对话框，用于引领您完成常见的任务，例如创建新文件和项目。

Workbench 一个包含透视图、菜单和工具栏的窗口。

工作区 文件系统上的一个指定区域，用于存储与项目相关的文件和文件夹。

A.2 键盘快捷键

本节列出 Workbench 中会用到的一些最常见键盘快捷键：

F3 单击跳转指令或 C/C++ 调用函数的某个汇编器标签并按 F3 可将编辑器焦点移至所选项的位置。

F10 与箭头键结合使用时可访问主菜单。

Alt+F4 退出 Workbench。

Alt+向左键

在导航历史记录中后退。

Alt+向右键

在导航历史记录中前进。

Ctrl+; ARM 汇编器编辑器附带的快捷键，用于向活动文件中的选定代码块添加注释标记。

Ctrl+End 将编辑器焦点移至代码末尾。

Ctrl+Home 将编辑器焦点移至代码开头。

Ctrl+F 打开“Find/查找”或“Find/Replace/查找/替换”对话框以在活动编辑器中搜索代码。 有些编辑器是只读的，因此会禁用此功能。

Ctrl+F4 在编辑器视图中关闭活动文件。

Ctrl+F6 在编辑器视图中循环访问打开的文件。

Ctrl+F7 循环访问可用视图。

Ctrl+F8 循环访问可用透视图。

Ctrl+F10 与箭头键结合使用时可访问下拉菜单。

Ctrl+L 移至活动文件中的指定行。

Ctrl+Q 移至活动文件中上次编辑的位置。

Ctrl+空格键 在编辑器中为选定函数提供自动完成功能。

Shift+F10 与箭头键结合使用时可访问上下文菜单。

Ctrl+Shift+F

激活“Preferences/首选项”对话框中的代码样式设置并将其应用于活动文件。

Ctrl+Shift+L

打开一个列出所有键盘快捷键的小页面。

Ctrl+Shift+R

打开“Open resource/打开资源”对话框。

Ctrl+Shift+T

打开“Open Type/打开类型”对话框。

Ctrl+Shift+/

C/C++ 编辑器附带的快捷键，用于向活动文件中选定代码块的开头和末尾添加注释标记。

A.3 菜单和工具栏图标

本节列出可在 workbench 中使用的一些最常见的菜单和工具栏图标。 有关以下各表中没有列出的图标、标记和按钮的信息，请参阅动态帮助中的标准《Workbench 用户指南》(*Workbench User Guide*) 或《C/C++ 开发用户指南》(*C/C++ Development User Guide*)。

表 A-1 ARM 图标

按钮	说明	按钮	说明
	ARM		
	创建新的 ARM 项目		加载到 RealView Debugger 中
	打开 Flash 编程器配置对话框		打开 Flash 配置对话框
	导航到上一页		导航到下一页

表 A-2 透视图图标

按钮	说明	按钮	说明
	打开新透视图		C/C++ 透视图

表 A-3 视图图标

按钮	说明	按钮	说明
	打开 **Overview/概览**		打开 **What’s New/新增功能**
	打开 **Samples/示例**		打开 **Tutorials/教程**

表 A-3 视图图标 （续）

按钮	说明	按钮	说明
	打开 **Workbench**		显示下拉菜单
	最小化视图		最大化视图
	还原视图		关闭视图

表 A-4 运行和调试图标

按钮	说明	按钮	说明
	运行程序		运行外部工具
	调试程序		

表 A-5 编辑器图标

按钮	说明	按钮	说明
	保存活动文件		保存所有文件
	打印活动文件		关闭文件
	创建新配置		复制所选配置
	删除所选配置		折叠配置树

表 A-6 编辑器标记

按钮	说明	按钮	说明
	书签		断点标记
	任务标记		搜索结果
	错误标记		警告标记
	信息标记		

表 A-7 大纲图标

按钮	说明	按钮	说明
	隐藏字段		隐藏静态成员
	隐藏非公共成员		按字母顺序排序
	类		命名空间
	宏定义		枚举
	枚举数		变量
	受保护字段		私有字段
	公共字段		包含
	受保护方法		私有方法
	公共方法		结构
	类型定义		联合
	函数		

表 A-8 其他图标

按钮	说明	按钮	说明
	打开新建资源向导		打开新建项目向导
	打开新建文件夹向导		打开新建文件向导
	打开搜索对话框		显示上下文相关帮助
	打开导入向导		打开导出向导

表 A-9 导航图标

按钮	说明	按钮	说明
	后退		前进
	后退		前进
	打开帮助说明页		将目录与活动页同步
	为活动页添加书签		打印活动页

表 A-10 帮助目录图标

按钮	说明	按钮	说明
	显示所有文档的列表		显示上次搜索的文档的列表

表 A-10 帮助目录图标 （续）

按钮	说明	按钮	说明
	显示指向相关主题的上下文相关帮助链接列表		显示所有书签的列表
	最大化框架		还原框架
	将目录与活动页同步		