RealView® 编译工具

4.0 版

要点指南

ARM DUI 0202IC

RealView 编译工具
要点指南

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	更改
2002 年 8 月	A	非保密	1.2 版
2003 年 1 月	B	非保密	2.0 版
2003 年 9 月	C	非保密	RealView Developer Suite 2.0.1 版
2004 年 1 月	D	非保密	RealView Developer Suite 2.1 版
2004 年 12 月	E	非保密	RealView Developer Suite 2.2 版
2005 年 5 月	F	非保密	RealView Developer Suite 2.2 SP1 版
2006 年 3 月	G	非保密	RealView Development Suite 3.0 版
2007 年 3 月	H	非保密	RealView Development Suite 3.1 版
2008 年 9 月	I	非保密	RealView Development Suite v4.0 4.0 版

所有权声明

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
RealView 编译工具
要点指南

前言

本前言介绍 《RealView 编译工具要点指南》。本前言分为以下几节：

关于本手册

本手册概要介绍 ARM® RealView® 编译工具 (RVCT)。

适用对象

本手册是为所有使用 RVCT 编写应用程序的开发者编写的。本手册假定您是一位经验丰富的软件开发人员。

使用本手册

本手册由以下章节和附录组成：

第 1 章 ***简介***

本章简要介绍了 RVCT。同时介绍了 RVCT 的组件及在线文档。

第 2 章 ***创建应用程序***

本章概述如何使用 RVCT 创建应用程序。

第 3 章 ***RVCT v4.0 与 RVCT v3.1 之间的差异***

本章介绍 RVCT 的最新版本和上一版本之间的差异。

附录 A ***关于早期版本***

本附录介绍 RVCT 各早期版本之间的差异。

本手册假定 ARM 软件安装在缺省位置。例如，在 Windows 上，这可能是 *volume*:\Program Files\ARM。引用路径名时，假定安装位置为 *install_directory*。例如， *install_directory*\Documentation\...。如果将 ARM 软件安装在其他位置，则可能需要更改此位置。

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、介绍特殊术语以及表示内部交叉引用和引文。
粗体	突出显示界面元素，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 公司的各种出版物，可提供有关 ARM 系列处理器开发代码的其他信息。

ARM 公司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 公司出版物

本手册包含了有关 RVCT 的一般信息。该套件中包含的其他出版物有：

- 《RealView 编译工具编译器用户指南》 (ARM DUI 0205)。本手册介绍 ARM 编译器 `armcc` 的基本功能以及特定于编译器的功能。另外还介绍 NEON™ 向量化编译器，并说明如何利用自动向量化功能。
- 《RealView 编译工具编译器参考指南》 (ARM DUI 0348)。本手册提供 ARM 编译器的参考信息，并介绍命令行选项。另外还提供了有关如何在该编译器中生成 C 和 C++ 的 ARM 实现的参考资料。
- 《RealView 编译工具库和浮点支持指南》(ARM DUI 0349)。本手册介绍了 ARM C 和 C++ 库、对 ISO 标准的遵循、自动调整与目标相关的函数以及特定于应用程序的要求。另外还介绍了 ARM 对浮点计算的支持。

- 《RealView 编译工具汇编器指南》 (ARM DUI 0204)。本手册提供了有关 ARM 汇编器 armasm 的参考和指导信息。
- 《RealView 编译工具链接器用户指南》(ARM DUI 0206)。本手册提供有关 ARM 链接器 armlink 的用户信息。另外还概述了分散加载。
- 《RealView 编译工具链接器参考指南》(ARM DUI 0381)。本手册提供有关命令行选项和控制文件的参考信息。另外还介绍了 *ARM 体系结构的基础平台 ABI* (BPABI) 以及 System V 共享库和可执行文件。
- 《RealView 编译工具实用程序指南》 (ARM DUI 0382)。本手册提供有关 ARM 库管理程序 armar 和 ARM 映像转换实用程序 fromelf 的命令行选项与可运行示例的信息。
- 《RealView 编译工具开发指南》(ARM DUI 0203)。本手册提供有关编写以 ARM 系列处理器为目标的代码的指导信息。
- 《ARM Workbench IDE 用户指南》(ARM DUI 0330)。本手册介绍如何使用集成开发环境 (IDE) 为 ARM 目标配置和生成项目。

有关基本标准、软件接口和 ARM 支持的标准的完整信息，请参阅 *install_directory*\Documentation\Specifications\...。

此外，有关与 ARM 产品相关的特定信息，请参阅下列文档：

- 《ARM 体系结构参考手册》 ARMv7-A 和 ARMv7-R 版 (ARM DDI 0406)
- 《ARM7-M 体系结构参考手册》 (ARM DDI 0403)
- 《ARM6-M 体系结构参考手册》 (ARM DDI 0419)
- 《ARM 体系结构参考手册》 (ARM DDI 0100)
- 您的硬件设备的 ARM 数据手册或技术参考手册

反馈

ARM Limited 欢迎提供有关 RealView 编译工具及其文档的反馈。

对 RealView 编译工具的反馈

如果您有关于 RVCT 的任何问题，请与您的供应商联系。为便于供应商快速提供有用的答复，请提供：

- 您的姓名和公司
- 产品序列号
- 工具的版本字符串，包括您所用工具的版本号和发布日期。
- 您运行的平台的详细信息，如硬件平台、操作系统类型和版本
- 能重现问题的一小段独立的程序
- 您预期发生和实际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

关于本手册的反馈

如果您发现本手册有任何错误或遗漏之处，请发送电子邮件到 errata@arm.com，并提供：

- 文档标题
- 文档编号
- 您要对其发表意见的页码
- 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

本章介绍 ARM® RealView® 编译工具 (RVCT) 及其软件组件和文档。本章分为以下几节：

1.1 关于 RealView 编译工具

RVCT 由一系列工具、支持文档和示例组成。这些工具可用于针对 ARM 系列处理器编写和生成应用程序。

可以使用 RVCT 来生成使用 C、 C++ 或 ARM 汇编语言编写的软件程序。

1.1.1 RVCT 的组件

本节概述 RVCT 组件。

开发工具

以下开发工具随 RVCT 一起安装：

armcc ARM 编译器。它可编译 C 和 C++ 代码。

armasm ARM 和 Thumb® 汇编器。该汇编器汇编 ARM 和 Thumb 汇编语言源代码。

armlink ARM 链接器。它可将一个或多个对象文件的内容与一个或多个对象库的选定部分相结合，生成一个可执行程序。

Rogue Wave C++ 库

Rogue Wave 库提供标准 C++ 库的实现。有关 Rogue Wave 库的详细信息，请参阅 CD ROM 上的 HTML 文档。

C++ 库 ARM C++ 库提供了：

- 编译 C++ 时使用的辅助函数
- Rogue Wave 库不支持的附加 C++ 函数

C 库 ARM C 库按照 C 和 C++ 标准中的定义提供了库功能的实现。有关详细信息，请参阅《库和浮点支持指南》中第 2-2 页上的“关于 C 和 C++ 库”。

C 微型库 ARM C 微型库 (Microlib) 提供了高度优化的函数集。这些函数可用于必须在极少量内存环境下运行的深层嵌入式应用程序。有关详细信息，请参阅《库和浮点支持指南》中的第 3 章 “C 微型库”。

fromelf ARM 映像转换实用程序。该实用程序也可生成有关输入映像 （例如反汇编及其代码和数据大小）的文本信息。

armar ARM 库管理程序。它可使多组 ELF 格式对象文件集中到一起并保留在档案或库中。可将这样的库或档案传给链接器，以替代多个 ELF 文件。也可以将该档案分发给第三方，以进行进一步的应用程序开发。

注意

RealView Development Suite (RVDS) 支持 64 位 Linux 平台，但 RVCT 的设计还不能利用此功能。RVCT 的文件 I/O 例程使用文件大小适合 32 位 (**signed**) **int** 的标准系统调用。这意味着最大映像大小限制为 2GB，即使在 64 位平台或内存大于 2GB 的计算机上进行构建也是如此。如果超过这个大小，链接器将报告一条错误消息，指示内存不足。这可能令人不解，因为有足够的内存但应用程序却无法访问。

标准遵从性

RVCT 符合以下标准。在每种情况中，都注明了遵从的程度：

ar UNIX 样式的对象代码档案，由 `armar` 生成，`armlink` 使用。`armar` 可列出和提取多数 `ar` 格式的对象代码档案，`armlink` 可使用由另一个档案实用程序创建的 `ar` 格式档案，只要该档案包含符号表成员即可。

DWARF 3 RVDS 中的所有工具都支持 DWARF 3 调试表 （DWARF 调试标准第 3 版）。

DWARF 2 RVCT 中的所有工具和 ARM 提供的 ELF DWARF 2 兼容调试器（例如 RealView Debugger）都支持 DWARF 2 调试表。

ISO C ARM 编译器接受 ISO C 1990 和 1999 源代码作为输入。有关详细信息，请参阅 《编译器用户指南》中的*源语言模式*。

ISO C++ ARM 编译器接受 ISO C++ 2003 源代码作为输入。

ELF ARM 工具可生成 ELF 格式的可重定位的和可执行的文件。`fromelf` 实用程序可将 ELF 文件转换为其他格式。

注意

DWARF 2 和 DWARF 3 标准在某些地方不明确，例如调试帧数据。这意味着无法保证第三方调试器能够使用 ARM 代码生成工具所生成的 DWARF，也无法保证 RealView Debugger 能够使用第三方工具所生成的 DWARF。

符合 ARM 体系结构的 ABI （基本标准）

ARM 体系结构的*应用程序二进制接口* (ABI) 是一个标准集。其中有些标准是开放的。有些是 ARM 体系结构特有的。这些标准控制基于 ARM 的执行环境（从裸机到 ARM Linux 等主流操作系统）中的二进制代码和开发工具的互操作。

只要符合此标准，则来自不同生成器的 ARM 和 Thumb 对象及对象库就可以协同工作。

《ARM 体系结构的 ABI （基本标准）》 (BSABI) 由一系列规范组成，
其中包括：

AADWARF 《ARM 体系结构 DWARF 标准》。此 ABI 使用 DWARF 3 标准来控制调试数据在对象生成器与调试器之间的交换。

AAELF 《ARM 体系结构 ELF 标准》。基于一般 ELF 标准，用于控制可链接和可执行文件在生成者与使用者之间的交换。

AAPCS 《ARM 体系结构的过程调用标准》。用于管理运行时各函数之间的控制和数据交换。RVCT 所支持的每个主要执行环境类型都有一个 AAPCS 变体。

BPABI 《ARM 体系结构的基本平台 ABI》。用于控制静态链接器所生成的可执行文件和共享对象文件的格式和内容。它使用链接后处理支持平台特定的可执行文件。它提供用于派生平台 ABI 的基本标准。

CLIBABI 《ARM 体系结构的 C 库 ABI》。定义 C 库的 ABI。

CPPABI 《ARM 体系结构的 C++ ABI》。基于一般 C++ ABI （最初是为 IA-64 开发的），用于控制独立 C++ 编译器之间的交互操作。

EHABI 《ARM 体系结构的异常处理 ABI》。定义异常的引发和处理方式中与语言无关及特定于 C++ 的方面。

RTABI 《ARM 体系结构的运行时 ABI》。控制哪些独立生成的对象可通过浮点和编译器辅助函数的支持假定其执行环境。

有关 ARM 支持的基本标准、软件接口以及其他标准的详细信息，请参阅 *install_directory*\Documentation_Specifications_4.0\PDF。

有关最新发行版本的详细信息，请访问 http://www.arm.com。

如果要从早期版本升级到最新版 RVCT，请确保使用最新版的 ARM 规范。

支持软件

要在仿真环境下或在基于 ARM 内核的硬件上调试程序，请使用合适的调试器，例如 RealView Debugger。它符合 ELF、 DWARF 2 和 DWARF 3 标准，由 GCC v3.4 或 RVCT v2.2 及更高版本生成。

要在仿真环境下调试程序，请使用 RealView ARMulator® ISS 或*指令集系统模型* (ISSM) 支持软件。RealView Armulator ISS 是随 RVDS 一起提供的*指令集仿真器* (ISS)。它与调试器进行通信，并可在运行调试器的主机上或调试器的远程系统上运行。有关详细信息，请参阅 *《RealView ARMulator ISS 用户指南》*。

本次新版提供了 Cortex™ 处理器的仿真器模型。可通过 RealView Debugger 中的 ISSM 调试接口访问这些模型。

代码示例

本手册引用了 RVDS 随附的示例，这些示例位于示例目录 *install_directory*\RVDS\Examples 中。有关所提供示例的汇总，请参阅《RealView Development Suite 入门指南》。

1.2 RVCT 使用的环境变量

表 1-1 显示了 RVCT 使用的环境变量。

表 1-1 RVCT 使用的环境变量

环境变量	设置
ARMROOT	安装根目录 (*install_directory*)。缺省为 C:\Program Files\ARM。
ARMLMD_LICENSE_FILE	ARM RealView 许可证文件的位置。有关此环境变量的信息，请参阅 《ARM 工具 FLEXnet 许可证管理指南》。
RVCT40_ASMOPT	要在常规 makefile 之外使用的其他 ARM 汇编器选项。例如： --licretry 列出的选项出现在 makefile 中为 armasm 命令指定的所有选项之前。因此，在 makefile 中指定的任何选项都可能覆盖在此环境变量中列出的选项。
RVCT40_CCOPT	要在常规 makefile 之外使用的其他 ARM 编译器选项。例如： --licretry 列出的选项出现在 makefile 中为 armcc 命令指定的所有选项之前。因此，在 makefile 中指定的任何选项都可能覆盖在此环境变量中列出的选项。
RVCT40_FROMELFOPT	要在常规 makefile 之外使用的其他 ARM fromelf 选项。例如： --licretry 列出的选项出现在 makefile 中为 fromelf 命令指定的所有选项之前。因此，在 makefile 中指定的任何选项都可能覆盖在此环境变量中列出的选项。
RVCT40_LINKOPT	要在常规 makefile 之外使用的其他 ARM 链接器选项。例如： --licretry 列出的选项出现在 makefile 中为 armlink 命令指定的所有选项之前。因此，在 makefile 中指定的任何选项都可能覆盖在此环境变量中列出的选项。
RVCT40BIN	RVCT 程序可执行文件： *install_directory*\RVCT\Programs\...\win_32-pentium
RVCT40INC	ARM 编译器包含以下文件： *install_directory*\RVCT\Data\...\include\windows
RVCT40LIB	ARM 编译器库文件： *install_directory*\RVCT\Data\...\lib
RVDS_PROJECT	标识项目模板目录。
RVDS_PROJECT_WORKDIR	标识项目工作目录。

1.3 获取详细信息

根据安装的不同，完整的文档套件将以基于浏览器的 HTML 格式以及 PDF 格式提供。

注意

《RealView Development Suite 入门指南》中提供了 RVDS 文档中所用 ARM 术语的术语表。

如果安装了文档套件，可使用以下某一方法访问该文档：

- 根据平台的不同，若要查看文档套件，请执行以下操作：
  - 在 Windows 上，请选择：
    开始 → 所有程序 → ARM → Help viewer v1.0
  - 在 Red Hat Linux 上，请选择：
    开始菜单 → 程序 → ARM → Help viewer v1.0

  这将显示一个单独的查看器，您可在其中：
  - 查看 HTML 格式的 RVDS 文档
  - 对所有文档或部分文档执行文本搜索
  - 访问每个文档的对应 PDF 文件

  注意

  在从独立查看器中查看 PDF 文档时，无法搜索所有 PDF 文档。

- 根据平台的不同，若要查看 PDF 文档，请执行以下操作：
  - 在 Windows 上，请选择：
    开始 → 所有程序 → ARM → RealView Development Suite v4.0 → RVDS v4.0 Documentation Suite/RVDS v4.0 文档套件
  - 在 Red Hat Linux 上，请选择：
    开始菜单 → 程序 → ARM → RealView Development Suite v4.0 → RVDS v4.0 Documentation Suite/RVDS v4.0 文档套件

  这将显示一个 PDF 文档，其中包含指向 PDF 格式 RVDS 文档的链接。您还可以对所有 PDF 文档执行文本搜索。

此外，在所有支持的平台上还提供 Rogue Wave C++ 库的 HTML 格式文档。此文档是标准安装模式缺省安装的。有关详细信息，请参阅第1-8 页的*Rogue Wave 文档*。

1.3.1 Rogue Wave 文档

RVCT 的 Rogue Wave 标准 C++ 库的手册在产品 CD ROM 中以 HTML 文件形式提供。可以使用标准 Web 浏览器查看这些文件。例如，选择文件 *install_directory*\Documentation\RogueWave\1.0\release\stdref\index.htm 将显示 Rogue Wave 的 HTML 文档。请参阅图 1-1，其中 *install_directory* 为 D:\ARM。

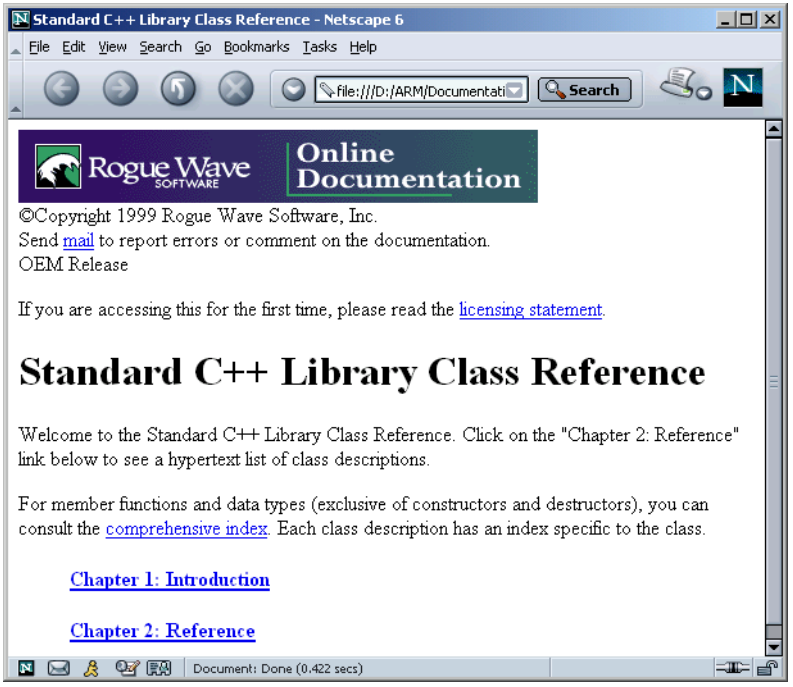

图 1-1 Rogue Wave HTML 文档

第 2 章
创建应用程序

本章介绍如何使用 ARM® RealView® 编译工具创建应用程序。本章分为以下几节：

2.1 使用 ARM 编译工具

典型的应用程序开发可能涉及：

- 主应用程序的 C/C++ 源代码 (armcc)
- 近硬件组件的汇编源代码 (armasm)，如中断服务例程
- 将所有对象链接在一起，以生成映像 (armlink)
- 将映像转换为纯二进制、 Intel Hex 以及 Motorola-S 格式的闪存格式 (fromelf)

图 2-1 演示了如何链接 ARM 编译工具以开发典型的应用程序。

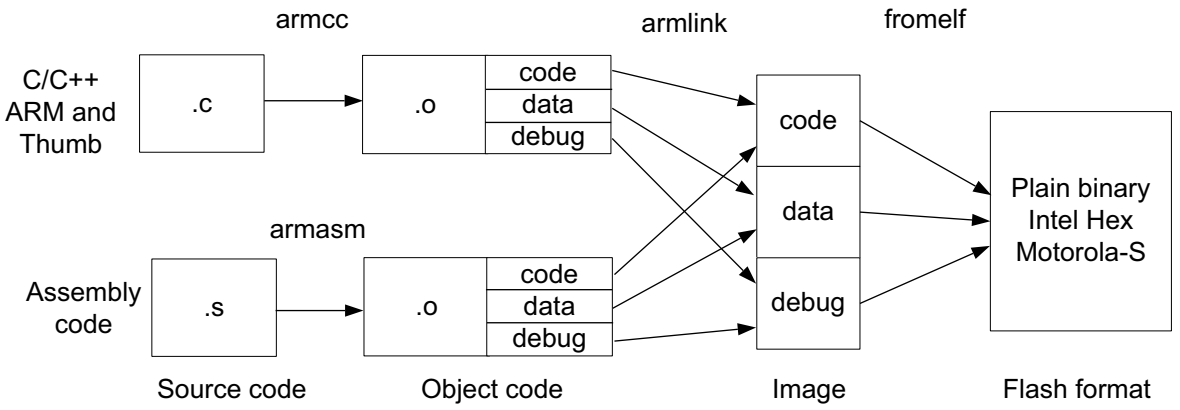

图 2-1 典型的工具使用流程图

2.2 使用 ARM 编译器

ARM 编译器 armcc 可将 C 和 C++ 源代码编译为 ARM 和 Thumb 代码。

通常，可按如下方式调用 ARM 编译器：

armcc [*options*] *ifile_1 ... ifile_n*

可以指定一个或多个输入文件。

有关 ARM 和 Thumb 的编译，以及如何根据您指定的文件扩展名调整编译器启动配置的详细信息，请参阅 《编译器用户指南》中的第 2 章 *ARM 编译器使用入门*。

2.2.1 生成 Dhrystone 示例

示例目录下安装了一系列应用程序的示例 C 源代码。每个示例都有一个 readme.txt 文件，该文件介绍示例代码以及如何编译该代码。

例如，Dhrystone 基准程序的源代码安装在示例目录中的 ...\dhrystone 之下。这个程序可用于评估系统的整数处理性能。

编译 Dhrystone 示例：

1. 使用以下命令编译 C 文件 dhry_1.c 和 dhry_2.c：

   ```
   armcc -c -W --debug -O3 -Otime --no_inline --no_multifile -DMSC_CLOCK \
       dhry_1.c dhry_2.c
   ```

   以下是常用选项：

   -c　　指示编译器只编译而不链接。

   --debug　　指示编译器为源代码级调试添加调试表。

   -O3　　指示编译器应用最高优化生成代码。

   -Otime　　指示编译器优化代码以提高速度，而不是节省空间。

   --no_inline 和 --no_multifile 选项是保留 Dhrystone 基准程序的主旨功能所必需的选项：

   - --no_inline 是必需的选项，用于禁用函数内联，因为 Dhrystone 要求不合并过程
   - --no_multifile 是必需的选项，用于禁用多文件编译，因为 Dhrystone 要求分别编译两个源文件。

   在生成 Dhrystone 编译文件时还会用到下列选项：

   -W　　指示编译器禁用所有警告。

-DMSC_CLOCK 指示编译器使用 C 库函数 clock() 进行计时度量。

有关编译器选项的详细信息，请参阅 《编译器参考指南》中的第 2 章 *编译器命令行选项*。

注意

请注意 --arm 是缺省编译器选项。有关详细信息，请参阅*编译生成 ARM 代码*。

2. 将文件链接在一起，请参阅第2-6 页的*使用 ARM 链接器*
3. 使用兼容的调试器 （例如 RealView Debugger）加载并运行映像。

有关 dhry_1.c 和 dhry_2.c 的内容以及如何计算 Dhrystone 性能的信息，请参阅该示例的 readme.txt 文件。

2.2.2 编译生成 ARM 代码

下列编译器选项用于生成 ARM 代码：

--arm 指示编译器优先生成 ARM 代码 （优先于 Thumb 代码）。但是，#pragma thumb 会覆盖此选项。这是缺省编译器选项。

--arm_only 强制编译器只生成 ARM 代码。编译器就当目标体系结构中没有 Thumb 一样进行工作。将忽略所有 #pragma thumb 声明。

另请参阅

- 《编译器用户指南》中第4-65 页的*#pragma thumb*
- 《编译器参考指南》中第2-7 页的*--arm*
- 《编译器参考指南》中第2-14 页的*--arm_only*

2.2.3 编译生成 Thumb 代码

要生成 Thumb 版本，请使用：

```
armcc --thumb ...
```

其中：

`--thumb` 指示编译器优先生成 Thumb 代码（优先于 ARM 代码）。但是，`#pragma arm` 会覆盖此选项。

另请参阅

- 《编译器用户指南》中第4-51 页的*#pragma arm*
- 《编译器参考指南》中第2-107 页的*--thumb*

2.3 使用 ARM 链接器

该链接器将一个或多个对象文件的内容与一个或多个对象库的选定部分合并起来，生成一个映像或对象文件。

通常，可如下调用 ARM 链接器：

```
armlink [options] file_1 ... file_n
```

有关详细信息，请参阅《链接器用户指南》中的第 2 章 *ARM 链接器使用入门*。

2.3.1 链接 Dhrystone 示例

对于 Dhrystone 示例程序，请使用下面的命令链接对象文件：

```
armlink dhry_1.o dhry_2.o -o dhrystone.axf --info totals
```

其中：

`-o` 将输出文件指定为 dhrystone.axf。

`--info totals` 指示链接器显示输入对象和库的代码和数据大小的总和。

2.4 使用 ARM 汇编器

使用 ARM 汇编器 (armasm) 的基本语法为：

```
armasm [options] inputfile
```

例如，若要汇编名为 myfile.s 的文件中的代码，并在结果对象文件中包含调试信息，请键入：

```
armasm --debug myfile.s
```

这会生成名为 myfile.o 的对象文件。

有关选项和语法的详细，请参阅 《编译器指南》中第3-2 页的*命令语法*。

2.4.1 从汇编器源代码生成示例

示例汇编语言代码安装在示例目录下。这些示例都有一个 readme.txt 文件，用以介绍这些示例以及如何编译它们。例如，有一个简单的程序 word.s，该程序的代码安装在示例目录中的 ...\asm 之下。

生成该示例：

1. 使用以下命令汇编源文件：
   ```
   armasm --debug word.s
   ```
2. 使用以下命令链接该文件：
   ```
   armlink word.o -o word.axf
   ```
3. 使用兼容的调试器 （例如 RealView Debugger）加载并测试映像。

   逐步调试该程序并检查寄存器，查看它们如何变化。有关如何执行此操作的详细信息，请参阅调试器文档。

2.5 使用 fromelf

ARM fromelf 实用程序的功能包括：

- 将 ELF 可执行格式的可执行映像转换为其他文件格式
- 控制输出文件中的调试信息
- 反汇编 ELF 映像或 ELF 对象文件
- 保护映像中的 IP 和传递到第三方的对象
- 打印有关 ELF 映像或 ELF 对象文件的信息

有关详细信息，请参阅 《实用程序指南》中的第 2 章 *使用 fromelf*。

2.5.1 使用 fromelf 示例

下面的示例演示如何使用 fromelf：

```
fromelf --text -c -s --output=outfile.lst infile.axf
```

创建一个纯文本输出文件，其中包含 ELF 映像的反汇编代码和符号表。

```
fromelf --bin --16x2 --output=outfile.bin infile.axf
```

针对内存配置为两个寄存器组 16 位内存宽度的目标系统创建两个二进制格式的文件 （outfile0.bin 和 outfile1.bin）。

最后一个示例的输出文件适合于直接写入 Flash 设备。

2.6 使用 ARM Workbench IDE

使用 ARM Workbench IDE，您可以通过图形用户界面来管理软件开发项目。ARM Workbench 提供了完全集成的 IDE，该 IDE 将软件开发与所有 RealView 工具的编译和调试技术结合在一起。Workbench 随附提供的示例包含一些可供您试用的完全可运行的项目。要使用这些项目，必须启动 Workbench，并将这些示例导入您的工作空间。有关详细信息，请参阅 《ARM Workbench IDE 用户指南》。

第 3 章
RVCT v4.0 与 RVCT v3.1 之间的差异

本章介绍 ARM® RealView® 编译工具 (RVCT) v4.0 和 RVCT v3.1 之间的主要差异。本章分为以下几节：

有关 RVCT 早期版本之间的差异，请参阅附录 A *关于早期版本*。

3.1 RealView 编译工具 v4.0 概述

RVCT v4.0 和 RVCT v3.1 之间最重要的差异是：

- 对 Cortex™-A9 （仅限 RVDS Professional 版）和 Cortex-R4F 处理器的支持。若要查看支持的体系结构和处理器的完整列表，请使用 `--cpu=list` 命令行选项。
- 对使用 RVCT 和 CodeSourcery 工具生成 Linux 应用程序的增强支持。
- 增强的符号可见性。
- 增强的 `fromelf` 功能。
- 链接器中增强的调用图功能。
- 生成适合预链接的映像。
- 可识别体系结构的反汇编。
- 对 Cortex™-M3 处理器的初步位处理操作支持。
- 增强的 Cortex™-M1 处理器浮点性能。
- 增强的代码优化。
- Profiler 引导的优化。
- 链接时代码生成。

3.2 RVCT v4.0 中文档的更改

RVCT v4.0 的文档有以下更改：

- 《链接器和实用程序指南》已拆分成以下手册：
  - *《链接器用户指南》*
  - *《链接器参考指南》*
  - 《实用程序指南》
- 《NEON™ 向量化编译器指南》已并入 《编译器用户指南》和 《编译器参考指南》。
- 《开发指南》已重新编排结构和更新，以反映最新的 ARM 体系结构和处理器。
- `--licretry` 选项已记载到文档中。ARM 汇编器、ARM 编译器、ARM 链接器和 `fromelf` 实用程序都支持该选项。
- 所有文档都已更新，以反映相关工具中的功能。

请参阅本章的其余部分，详细了解这些工具在本版本中的改动情况。

3.3 RVCT v4.0 中 ARM 编译器的改动

RVCT v4.0 中的编译器有以下改动：

- 支持 Cortex-A9 和 Cortex-R4F 处理器 （使用 `--cpu=Cortex-A9` 和 `--cpu=Cortex-R4F` 开关）。请参阅 《编译器参考指南》中第2-2 页的*命令行选项*。
- 添加了很多新的内在函数。请参阅 《编译器用户指南》中第4-2 页的*内在函数*。
- 添加了 `--arm_only` 命令行选项，该选项强制编译器仅输出 ARM 代码，忽略所有 `#pragma thumb` 声明。

  请参阅 《编译器参考指南》中第2-2 页的*命令行选项*。
- 添加了以下命令行选项，以支持生成 Linux 应用程序：
  - `--arm_linux`
  - `--arm_linux_config_file`
  - `--arm_linux_configure`
  - `--arm_linux_paths`
  - `--configure_cpp_headers`
  - `--configure_extra_includes`
  - `--configure_extra_libraries`
  - `--configure_gcc`
  - `--configure_gld`
  - `--configure_sysroot`
  - `--shared`
  - `--translate_g++`
  - `--translate_gcc`
  - `--translate_gld`
- 为 `--fpu` 命令行选项增加了以下选项：
  - `vfpv3_fp16`
  - `vfpv3_d16`
  - `vfpv3_d16_fp16`
  - `softvfp+vfpv3_fp16`
  - `softvfp+vfpv3_d16`
  - `softvfp+vfpv3_d16_fp16`

  相应的预定义宏也已添加。请参阅 《编译器用户指南》中第5-28 页的*使用浮点算法*。

- 已添加 __attribute__((bitband)) 类型属性，以支持 Cortex-M3 处理器上的位操作处理。请参阅 《编译器参考指南》中第4-37 页的*类型属性*。
- 增强的 Cortex-M1 处理器浮点性能。单精度算法至少快 6 倍，双精度算法至少快 3.5 倍。
- 用于显式或隐式标有 dllexport 的对象和函数的 ELF 符号可见性已更改。之前在 RVCT v3.1 STV_DEFAULT （可预占）节中定义的符号现在在 RVCT v4.0 STV_PROTECTED （导出的不可预占）节中定义。但是，COMDAT 节中定义的符号没有移动，仍在 STV_DEFAULT 节中定义。

  --no_hide_all 编译器选项仍然将符号的可见性设为 STV_DEFAULT，而不是 STV_HIDDEN，因此这将不再等同于导出它们。
- 新的 --retain 编译器选项可用来限制编译器执行的代码转换。例如，您可以防止误删未使用的内联函数。此外还提供了以下属性：
  - __attribute__((notailcall))
  - __attribute__((nomerge))

  请参阅 《编译器参考指南》中第2-98 页的*--retain=option*。

  请参阅 《编译器参考指南》中第4-28 页的*函数属性*。
- Profiler 引导的优化利用了 ARM Profiler 生成的应用程序性能信息。可以使用新的 --profile 选项将性能信息输入编译器和链接器中，以生成大小更小性能却更快的代码。
- 链接时代码生成是使用新的 --ltgc 编译器和链接器选项实现的，它提供了以下优化：
  - 跨模块内联以提高性能
  - 共享基址以缩小代码大小

  有关详细信息，请参阅 《编译器用户指南》中第5-16 页的*函数内联*。
- 增加的其他命令行选项有：
  - --compatible
  - --device
  - --fp16_format
  - --library_interface=rvct_c90
  - --licretry
- 为兼容 GNU 库头文件，增加了一些函数。请参阅 《编译器参考指南》中第4-104 页的*GNU 内置函数*。

3.4 RVCT v4.0 中对库支持的改动

宏和标准库之间的功能差异已减小。此外还支持宽字符 IO。有关详细信息，请参阅 《库和浮点支持指南》。

3.5 RVCT v4.0 中 ARM 链接器的改动

RVCT v4.0 中的链接器有以下改动：

- 增加了以下命令行选项：

  `--[no_]add_needed`

  `--arm_only`

  `--arm_linux`

  `--[no_]combreloc`

  `--device`

  `--filtercomment`

  `--info=visibility`

  `--no_largeregions`

  `--licretry`

  `--ltcg`

  `--max_open_files`

  `--[no_]muldefweak`

  `--[no_]prelink_support`

  `--profile`

  `--section_index_display=`*`type`*

另请参阅：

- 《链接器用户指南》中第2-6 页的*使用命令行选项*
- 《链接器参考指南》中的第 2 章 *链接器命令行选项*

3.6 RVCT v4.0 中 ARM 汇编器的改动

RVCT v4.0 中的汇编器有以下改动：

- 增加了以下 NEON™ 和 VFP 指令：
  - VCVT，带半精度扩展
  - VCVTB 和 VCVTT，带半精度扩展

  请参阅 《汇编器指南》中第5-39 页的*NEON 通用数据处理指令*和第5-97 页的*VFP 指令*。
- 增加了 PLDW 指令以支持 Cortex-A9。有关详细信息，请参阅 《汇编器指南》中第4-23 页的*PLD*、 *PLDW 和 PLI*。
- 以下指令现在支持符号类型：
  - EXPORT
  - EXTERN
  - GLOBAL
  - IMPORT
- 增加了以下命令行开关：
  - --arm_only
  - --cpreproc
  - --device
  - --licretry

  有关详细信息，请参阅 《汇编器指南》中第 3 章 *汇编器参考*。

另请参阅 《汇编器指南》中第5-15 页的*一般信息*。

3.7 RVCT v4.0 中 fromelf 实用程序的改动

RVCT v4.0 中的 fromelf 实用程序添加了以下选项：

- --base
- --bincombined
- --bincombined_base
- --bincombined_padding
- --cad
- --cadcombined
- --compare
- --cpu
- --datasymbols
- --device
- --disassemble
- --emit
- --fpu
- --globalize
- --hide
- --hide_and_localize
- --ignore_section
- --ignore_symbol
- --info=instruction_usage
- --in_place
- --interleave
- --licretry
- --localize
- --privacy
- --qualify
- --reinitialize_workdir
- --relax_section
- --relax_symbol
- --rename
- --show
- --show_and_globalize
- --source_directory

- `--strip=symbols`
- `--strip=localsymbols`
- `--workdir.`

以前没有文档记载的选项 `--text -w` 现已记入文档。

支持识别体系结构的反汇编。

有关详细信息，请参阅 《实用程序指南》中的第 2 章 *使用 fromelf*。

3.8 RVCT v4.0 中不提倡使用的功能

RVCT v4.0 中不提倡使用以下功能：

- 不提倡使用 `--memaccess` 编译器选项。
- 不提倡使用以下 mathlib 函数：
  - `gamma()`
  - `gamma_r()`
  - `lgamma_r()`
  - `scalb()`
  - `significand()`
- 不提倡使用以下 Bessel 函数：
  - `j0()`
  - `j1()`
  - `jn()`
  - `y0()`
  - `y1()`
  - `yn()`
- ARM 汇编器选项 `-0`（大写，非小写）

3.9 RVCT v4.0 中不再使用的功能

RVCT v4.0 中不再使用以下功能：

- 不再使用以下工具：
  - armcpp
  - tcc
  - tcpp

  makefile 中对 armcpp、tcc 或 tcpp 的所有引用必须分别更改为 armcc --cpp、armcc --thumb 或 armcc --thumb --cpp。
- 不再使用通过 C 宏调整语言环境和 CTYPE 的做法。
- 不再使用 ARM 汇编器选项 -D （与 --depend 同义）。

3.10 RVCT v4.0 与旧对象和库的兼容性

只要没有使用 `--apcs /adsabi` 进行生成，并且使用了 RVCT v4.0 链接器和 C/C++ 库，则支持与 RVCT v2.*x* 和 v3.x 对象/库代码的向后兼容性。不保证向前兼容。

由于存在这些限制，ARM 强烈建议您使用 RVCT v4.0 重新生成包括所有用户或第三方提供的库在内的整个项目。这是为避免任何潜在的不兼容性，并充分利用 RVCT v4.0 提供的改进优化、增强功能和新增功能。

附录 A
关于早期版本

本附录介绍早期版本的 ARM® RealView® 编译工具 (RVCT) 之间的主要差异。

本附录分为以下几节：

A.1 RVCT v3.1 与 RVCT v3.0 之间的差异

本节介绍 RVCT v3.1 与 RVCT v3.0 之间的差异。

A.1.1 RVCT v3.1 概述

- RVCT v3.1 现在支持以下新的 `--cpu` 选项：
  - ARM v7 是不可识别的 ARM 体系结构。`--cpu=7` 表示对所有 ARMv7-A、ARMv7-R 和 ARMv7-M 体系结构都通用的功能。根据定义，`--cpu=7` 选项所具有的任何给定功能在所有 ARMv7-A、ARMv7-R 和 ARMv7-M 体系结构中都存在。
  - Cortex™ 处理器：

    ```
    --cpu=Cortex-R4
    --cpu=Cortex-M1
    --Cortex-A8NoNEON.
    ```

  - Marvell Feroceon 处理器 `--cpu=88FR`*xxx*。

  要查看支持的体系结构和处理器的完整列表，请使用 `--cpu=list`。有关详细信息，请参阅 《编译器参考指南》中第2-28 页的*`--cpu=list`*和第2-28 页的*`--cpu=name`*。
- RVCT v3.1 删除了 `--apcs=/adsabi` 选项。无法再编译 ADS 兼容的对象以及链接旧 ADS 对象和库。
- RVCT v3.1 提供完整的 C99 语言支持，复数和宽 I/O 除外。有关详细信息，请参阅 《编译器用户指南》中第5-41 页的*C99 的新功能*。
- RVCT v3.1 完全支持 NEON™ 技术。有关详细信息，请参阅 《编译器用户指南》中的第 3 章 *使用 NEON 向量化编译器* 和第4-10 页的*NEON 内在函数*。
- RVCT v3.1 提供了 C 微型库 (microlib) 作为标准 C 库的替代库，可用于适合在极少量内存环境下运行的深度嵌入式应用程序。Microlib 并不完全符合 ISO C 标准。有关详细信息，请参阅《库和浮点支持指南》中第 3 章 *C 微型库*。
- RVCT v3.1 提供汇编器宏来调整语言环境和 CTYPE 函数。有关详细信息，请参阅 《库和浮点支持指南》中第2-41 页的 *使用汇编器宏调整语言环境和 CTYPE*。

- RVCT v3.1 提供特定的编译器内在函数，有助于在 C 和 C++ 中轻松访问基于 ARM 体系结构的处理器的低级功能。有关详细信息，请参阅 《编译器用户指南》中第4-3 页的*指令内在函数*。
- RVCT v3.1 支持 ETSI 内在函数。有关详细信息，请参阅 《编译器用户指南》中第4-6 页的*ETSI 基本运算*。
- RVCT v3.1 支持 C55x 内在函数。有关详细信息，请参阅 《编译器用户指南》中第4-8 页的*TI C55x 内在函数*。
- RVCT v3.1 提供对调用图、分散加载文件和执行区节布局的增强支持。有关详细信息，请参阅 《链接器参考指南》中第2-7 页的*--[no_]callgraph*。
- RVCT v3.1 提供对可交付的映像和对象中 IP 的增强保护。有关详细信息，请参阅第A-9 页的*RVCT v3.1 中 fromelf 实用程序的改动*。
- RVCT v3.1 提供的一个选项可以控制去除文件路径中的冗余路径名信息。此选项可用于 ARM 编译器、链接器和汇编器。有关详细信息，请参阅 《链接器参考指南》中第2-45 页的*--[no_]reduce_paths*。
- RVDS v3.1 现在包含 Eclipse IDE 作为主要的项目管理工具。RVDS 的 Eclipse 插件提供综合性的配置面板，可设置 RVCT 的所有选项。有关详细信息，请参阅 《RealView Development Suite Eclipse 插件用户指南》。
- RVDS v3.1 已将 `--verbose` 命令行选项更改为将诊断输出发送到 `stdout`，而不是 `stderr`。如果要将此信息重定向到文件，则必须使用 `--list`，而不是 `--errors`。

A.1.2 RVCT v3.1 中不再使用的功能

请注意 RVCT v3.1 中的以下差异：

- RVCT v3.1 不支持 Solaris 平台。
- RVCT v3.1 不再支持 Red Hat Linux Enterprise v3 32 位主机平台。
- RVCT v3.1 不支持编译器选项 `--apcs=/adsabi`。
- RVCT v3.0 不提倡使用的功能在 RVCT v3.1 中不再使用。有关详细信息，请参阅第A-12 页的*RVCT v3.0 中不提倡使用的功能*。

A.1.3 RVCT v3.1 中不提倡使用的功能

请注意 RVCT v3.1 中的以下差异：

- 不提倡使用以下工具名称：
  - armcpp
  - tcc
  - tcpp
- 不提倡使用通过文件扩展名 （如 .ac 和 .tc）进行的指令集切换。
- 在针对 ARMv7-A、ARMv7-R 和 ARMv7-M 或更高版本的体系结构进行编译时，不提倡使用内联汇编器。对于 ARMv7-A 和 ARMv7-R，编译器会生成警告。对于 ARMv6-M 和 ARMv7-M，在生成 Thumb 代码时，编译器会生成错误。
- 不提倡使用以下选项：
  - --split_ldm
  - --memaccess
- 不提倡使用 C 宏调整语言环境和 CTYPE。
- 不提倡使用 fromelf 选项 --no_comment_section，文档中已不再包含该选项。应改用 --strip=comment 选项。
- 不提倡使用以下汇编器功能：
  - 在新代码中不提倡使用 VFP 向量模式，并且统一汇编语言不支持向量记号。若要使用向量记号，必须使用旧的 VFP 助记符。
  - 对于更新的 CPU，不提倡使用 CPSR 访问可在用户模式下访问的位。而应使用 APSR。
  - 对于 ARMv6T2 及更高版本，在 STC 和 STC2 中使用 PC 相对的寻址。
  - --checkreglist，改用 --diag_warning 1206。
  - 将 16 位 LDM 和 STM 与寄存器列表中的基址寄存器一起使用，并指定回写。

A.1.4 RVCT v3.1 中 ARM 编译器的改动

请注意 RVCT v3.1 中编译器的以下差异：

- `--cpu` 命令行选项支持新的体系结构和处理器。有关详细信息，请参阅第A-2 页的*RVCT v3.1 概述*。
- `--vectorize` 命令行选项直接从 C 或 C++ 代码生成 NEON™ 向量指令。有关详细信息，请参阅 《编译器参考指南》中第2-116 页的*`--[no_]vectorize`*。

  注意

  要使用此选项，您必须有 NEON 向量化编译器许可证。该许可证随 RVDS Professional 版一起提供。
- `--c99` 命令行选项提供完整的 C99 语言支持，复数和宽 I/O 除外。
- `--library_type=`*`lib`* 命令行选项选择相关运行时库。有关详细信息，请参阅《编译器参考指南》中第2-72 页的*`--library_type=lib`*。
- `--bss_threshold=`*`num`* 命令行选项控制小型全局 ZI 数据项布局。
- 控制优化诊断消息的 `--diag_suppress=optimizations` 和 `--diag_warning=optimizations` 命令行选项。
- 控制变长数组支持的 `--[no_]vla` 命令行选项。
- `--wchar16` 和 `--wchar32` 命令行选项更改 `wchar_t` 类型。
- RVCT v3.1 不支持 `--apcs=/adsabi`。
- `--[no_]reduce_paths` 选项控制在文件路径中去除冗余路径名信息。
- 提供 NEON™ 内在函数，以从 C 和 C++ 生成 SIMD 指令。
- 支持对选定的 Texas Instruments C55x 内在函数的仿真。
- 支持实现语音编码的 ETSI 基本操作。
- `__attribute__((section("`*`name`*`")))` 用于对与 `--autoat` 命令行选项结合使用的布局地址进行编码。
- 提供特定的内在函数，以便使用 C 和 C++ 轻松访问基于 ARM 体系结构的处理器的低级功能。

A.1.5 RVCT v3.1 中对库支持的改动

请注意 RVCT v3.1 中的库支持的以下差异：

- RVCT v3.1 有新的库命名约定。
- RVCT v3.1 提供的库支持可用于必须在极少量内存环境下运行的深层嵌入式应用程序。Microlib 引入了一组新函数，这些函数并不完全遵从标准，但是为尽量减少代码进行了高度优化。
- RVCT v3.1 提供汇编器宏来调整语言环境和 CTYPE 函数。

另请参阅：

- 《库和浮点支持指南》

A.1.6 RVCT v3.1 中 ARM 链接器的改动

请注意 RVCT v3.1 中链接器的以下差异：

- `--info summarysizes` 属性汇总映像代码和数据大小。
- `--cpu=`*name* 命令行选项为选定的 ARM 处理器或体系结构指定上限。
- `--fpu=`*name* 命令行选项为选定的 FPU 体系结构指定上限。
- `--bpabi` 命令行选项生成 BPABI 可执行文件。
- `--dll` 命令行选项生成 BPABI DLL。
- `--predefine="`*string*" 命令行选项可将命令传递到在分散文件第一行中指定的预处理器。
- `--[no_]reduce_paths` 选项控制在文件路径中去除冗余路径名信息。
- `--autoat` 命令行选项控制 `__at` 节到执行区的自动放置。
- `--verbose` 命令行选项将诊断输出发送到 `stout`。如果要将此信息重定向到文件，则必须使用 `--list`。有关详细信息，请参阅 《链接器参考指南》中第2-59 页的*--verbose*。
- 可通过以下方式将变量分配给 `__at` 节：使用 `__attribute__((section("`*name*`")))` 显式命令该节，或者使用 `__attribute__((at("`*name*`")))` 为您设置节的名称。有关详细信息，请参阅《链接器用户指南》中第5-9 页的*指定区和节地址的示例*。

- ALIGN *alignment* 属性用于在分散加载文件中对齐边界。
- 向执行区分配输入节时，使用 .ANY*num* 进行优先级排序。
- 执行区内容的属性：
  - ZEROPAD 属性在 ELF 文件中对节进行初始化
  - 设置填充字节的值的 PADVALUE 属性
  - 创建一个链接器生成的区，其中包含一个值的 FILL 属性。
- --pltgot=*type*、 --pltgot_opts=*mode* 和 --info pltgot 命令行选项生成与 BPABI 的不同寻址模式相对应的表
- --library_type=*lib* 命令行选项选择相关运行时库。有关详细信息，请参阅《编译器参考指南》中第2-72 页的*--library_type=lib*。
- 调用图选项：
  - --callgraph_file=*filename* 定义输出文件的名称
  - --callgraph_output=*fmt* 定义输出文件的类型
  - --cgfile=*opt* 控制调用图的内容
  - --cgsymbol=*type* 控制调用图的内容
  - --cgundefined=*type* 控制调用图的内容。
- RVCT v3.1 中，链接器解释相对基址加载区的方式稍有不同。它不再根据前一个加载区中的*零初始化* (ZI) 数据调整加载区的基址。

  示例 A-1 显示了一个包含重叠数据的分散文件。

示例 A-1 包含 LR2 的相对基址的分散文件

```
LR1 0x8000
{
    er_progbits +0
    {
        *(+RO,+RW) ; Takes space in the Load Region
    }
    er_zi +0
    {
        *(+ZI) ; Takes no space in the Load Region
    }
}
LR2 +0 ; Load Region follows immediately from LR1
{
    er_moreprogbits +0
    {
```

```
        file1.o(+RO) ; Takes space in the Load Region
    }
}
```

如果检测到重叠， RVCT v3.1 会生成错误消息。为了纠正此问题，分散文件中要有一个表达式来计算 LR2 的基址。示例 A-2 显示了更正后的分散文件。

示例 A-2 包含计算得到的 LR2 基址的分散文件

```
LR1 0x8000
{
    er_progbits +0
    {
        *(+RO,+RW) ; Takes space in the Load Region
    }
    er_zi +0
    {
        *(+ZI) ; Takes no space in the Load Region
    }
}
LR2 ImageLimit(er_zi) ; Set the address of LR2 to limit of er_zi
{
    er_moreprogbits +0
    {
        file1.o(+RO) ; Takes space in the Load Region
    }
}
```

A.1.7 RVCT v3.1 中 ARM 汇编器的改动

请注意 RVCT v3.1 中汇编器有以下差异：

- `--cpu` 命令行选项支持新的体系结构和处理器。有关详细信息，请参阅第A-2 页的*RVCT v3.1 概述*。
- `--library_type=lib` 命令行选项选择相关运行时库。
- `--[no_]reduce_paths` 选项控制在文件路径中去除冗余路径名信息。
- 将 APSR 用于 CPSR 的用户模式位。
- 在新代码中不提倡使用 VFP 向量模式，统一汇编语言不支持向量记号。
- 支持 Intel 无线 MMX2™ 指令。

- CODEALIGN 属性可在 AREA 指令中设置。
- --depend_format=*string* 可更改输出相关性文件的格式。
- --no_code_gen 选项可与 --depend 一起使用，不生成对象文件。
- UND 是一种可对未在体系结构方面定义的指令进行编码的新指令。

有关详细信息，请参阅 《汇编器指南》。

A.1.8 RVCT v3.1 中 fromelf 实用程序的改动

请注意 RVCT v3.1 中的以下差异：

- 更改节名称的 --privacy 命令行选项
- --strip 命令行选项保护可交付的映像和对象中的 IP。

另请参阅：

- 《实用程序指南》

A.2 RVCT v3.0 与 RVCT v2.2 之间的差异

本节介绍 RVCT v3.0 与 RVCT v2.2 之间的差异。

A.2.1 RVCT v3.0 中的一般性改动

RVCT v3.0 中有以下改动：

- RVCT v3.0 包括对两个 ARMv7 体系结构配置文件的支持：

  ARMv7-A

  支持 ARM 和 Thumb-2 指令集的系统的应用程序性能信息，具有 Thumb®-2EE，可用于虚拟 （基于 MMU 的）内存系统。

  ARMv7-M

  仅支持 Thumb-2 的微控制器配置文件。

  RVCT v3.0 支持 ARMv4 以后的所有 ARM 体系结构。ARMv4 之前的所有体系结构名称现已过时，将不再受到支持。

- RVCT v3.0 支持两个 ARM Cortex™ 处理器系列：
  - Cortex™-A8
  - Cortex™-M3

  要查看支持的体系结构和处理器的完整列表，请使用：

  ```
  armcc --cpu list
  ```

- RVCT v3.0 包含对 ARMv7 体系结构的两个组件的初步支持，即高级 SIMD 扩展 （也称为 NEON™ 技术）和 VFPv3。这表示大量的 SIMD （*单指令、多数据*）指令以及某些 VFPv3 指令现在都可用在 ARM 和 Thumb-2 指令集中。

  NEON™ 是针对信号处理应用程序和嵌入式处理器的 64/128 位混合 SIMD 技术。NEON 作为处理器的一部分来实现，但是它拥有自己的执行管道，以及有别于 ARM 的寄存器组。NEON 指令在 ARM 和 Thumb-2 中都可用。

  要查看支持的浮点体系结构的完整列表，请使用：

  ```
  armcc --fpu list
  ```

- RVCT v3.0 支持 ARMv7 的 Thumb-2 执行环境 (Thumb-2EE)。Thumb-2EE 指令集基于 Thumb-2，并更改和增加了一些内容，使其更适用于动态生成的代码。

  Thumb-2EE 扩展引入了新的指令集状态，即 ThumbEE 状态。在此状态下，该指令集几乎与 Thumb-2 指令集完全相同。但有些指令的行为有所修改，并提供了一些新指令。

- ARM 汇编器可用于汇编 Intel® 无线 MMX™ 技术指令来开发代码，例如用于 PXA270 处理器。
- 按照最新版本的 《ARM 体系结构的 ABI （基本标准）》 [BSABI] 的要求， RVCT v3.0 完全支持 DWARF 3 调试表 （DWARF 调试标准第 3 版）。

  DWARF 3 是缺省格式并且包括：
  - 完全支持调试 C++ 命名空间
  - 调试信息的大小的增加。

  在此版本的编译器和汇编器中：
  - 如果您不指定格式，则假定为 DWARF 3
  - 如果您指定 `--debug` (`-g`)，缺省情况下，生成 DWARF 3。

  为了向后兼容，仍然支持命令行选项 `--dwarf2`。
- ARM 编译器和链接器支持*线程局部存储* (TLS)，使程序可以使用多线程。提供两个新的关键字来支持 MPCore™ 上的 TLS：
  - `__declspec(thread)`
  - `__thread`.
- ARM 编译器现在提供高级标量优化，包括展开循环。当使用 `-O3` 进行编译时，会自动调用这些增强功能，这样，以少量的代码大小开销即可获得显著的性能优势。
- 在 RVCT v3.0 中，整数 （**`int`** 或 **`long long`**）除以零缺省将返回零。这和以前的行为不同，以前运行时的结果是终止程序，并显示错误消息。
- 在 RVCT v3.0 中，您可以更改错误和警告消息使用的语言。例如，要在基于英语的工作站上显示日语消息，请使用：

  ```
  --message_locale ja_JP
  ```
- `--show_cmdline` 选项以前只在 ARM 编译器中可用，现在可以在汇编器、链接器、`armar` 和 `fromelf` 中使用。使用此选项可以查看命令行是如何被处理的。命令以标准化方式显示，并展开所有 via 文件的内容。但使用此选项不会捕获命令行中的意外错误。
- RVCT v3.0 引入了以下环境变量：
  - `RVCT30_CCOPT`
  - `RVCT30_LINKOPT`
  - `RVCT30_ASMOPT`

  对每个 RVCT，此变量的值将相应地插入到命令行字符串之前。命令行上的参数可覆盖环境变量中指定的选项。

RVCT v3.0 中不再使用的功能

RVCT v3.0 中有以下改动：

- RVCT v2.2 中不提倡使用的所有功能和命令行选项在 RVCT v3.0 中不再使用。要查看这些功能和选项的列表，请参阅《编译器用户指南》中的 “附录 A：使用旧命令行选项”。
- 早期版本的 RVCT 支持旧的编译器选项，以便帮助您将消息选项迁移到新版本。但这些选项现在已不再使用，也不再受支持。有关详细信息，请参阅 《编译器用户指南》中的 “附录 A：使用旧命令行选项”。
- ARM 编译器不支持多行字符串。此行为以前在 GNU 模式中受支持，以便向后兼容。
- 不再支持以下特定于编译器的编译指令：
  - `check_printf_formats`、`check_scanf_formats` 和 `check_vprintf_formats`
  - `[no_]debug`
- 不再支持软件堆栈检查。在此版本中，不支持以下编译器和汇编器选项：
  - `--apcs /swst`
  - `--apcs /noswst`
  - `--apcs /swstna`

  以下内容不再可用：
  - 编译指示 `[no_]check_stack`
  - 预定义宏 `__APCS_SWST`

  不再提供库的软件堆栈检查的版本。

RVCT v3.0 中不提倡使用的功能

RVCT v3.0 中有以下改动：

- ARM 编译器支持使用选项 `--apcs /adsabi` 来编译与旧的 *ARM Developer Suite*™ (ADS) *应用程序二进制接口* (ABI) 兼容的代码。RVCT 3.0 中不提倡这样做，而在 RVCT 3.1 中已取消这种做法。
- 以下是不提倡使用的汇编器选项， RVCT 3.1 中已取消这些选项：
  - `--no_cache`
  - `--no_regs` （改为使用 `--regnames=none`）
  - `--checkreglist` （改为使用 `--diag_warning 1206`）

- 不提倡将节属性限定符 （RO、 RW、 ZI、 DBG）与链接器选项 --remove 一起使用，将在以后的版本中删除。
- C++ 配置选项 --dll_vtbl 已由新的 --export_all_vtbl 选项取代。不提倡使用 --dll_vtbl，并且以后将不再支持该选项。
- 不提倡使用 --memaccess -UL41，将在以后的版本中删除。 --memaccess -UL41 被 --no_unaligned_access 替代。
- 编译器选项 --depend_format=unix 替代了 --unix_depend_format。此选项不提倡使用，将在以后的版本中删除。
- 在 RVCT v3.0 中， fromelf --text 的语法有所不同。以下形式不提倡使用，将在以后的版本中删除：
  - fromelf --text=*xx*
  - fromelf --text/*xx*
- ARM 不建议使用 __user_stack_slop 函数。此选项不提倡使用，将在以后的版本中删除。

A.2.2 RVCT v3.0 中 ARM 编译器的改动

RVCT v3.0 中有以下改动：

- 作为 RVCT v3.0 中新的可移植性功能的一部分，新的编译器选项 --library_interface 指定将编译器输出和 RVCT 库或任何符合 AEABI 的库一起使用。例如，使用 --library_interface=aeabi_glibc 将指定输出与符合 AEABI 的某版本 GNU C 库一起使用。
- ARM 编译器现在支持线程局部存储 (TLS)，使程序可以使用多线程。使用 TLS，每个线程可以修改全局变量，但更改仅对线程可见。提供两个关键字：
  - __declspec(thread) 声明变量是线程局部变量，并具有线程存储时限
  - __thread 相当于 __declspec(thread)。
- ARMv7 之前的 ARM 处理器使用 SWI 或 SVC 指令来进行半主机调用。不过，如果为 Cortex™-M3 处理器进行编译，则会使用 BKPT 指令实现半主机。

  使用新的 __semihost() 内在函数，您可以从与目标体系结构无关的 C 或 C++ 进行半主机调用。
- ARM 编译器提供内在函数来为 Cortex™-A8 处理器生成 ARM 和 Thumb 状态下的代码，从而支持 NEON™ 技术。

- ARM 编译器支持 GCC builtin 函数 __builtin_expect，当以反馈为导向的优化不切实际时提供跳转预测信息。此函数在 GNU 模式和 ARM 模式下可用。
- ARM 编译器包括 --info totals 选项，用于显示对象大小。使用此选项可以查看编译的对象中代码和数据 （RO 数据、 RW 数据、 ZI 数据和调试数据）的大小。
- 编译器包括可在 Windows 上使用的新的 --depend_format=*string* 选项。此选项将输出相关性文件的格式更改为 UNIX 风格，以便与某些 UNIX make 程序兼容。
- ARM 编译器现在可以识别 C++ 文件的 .cc 扩展名。当有以 .c 开头的无法识别的扩展名时，将发出警告，例如 *filename*.cmd。
- 作为 ARM 对 GNU 编译器扩展的支持的一部分， armcc 现在在使用 --gnu 选项时接受 C 和 C++ 语言的条件。如果结果与测试相同，则条件语句中的中间操作数可以忽略，例如：

  ```
  i ? i : j
  ```

  如果测试以某种方式修改了值，则这是尤其有用的，因为只需计算一次 i。
- RVCT v3.0 包含对 POSIX 函数 wcstombs() 的编译器和库支持，以便转换来自一个数组的多字节字符序列，然后将这些字符存储到其他数组中。
- ARM 编译器支持选项 --unaligned_access （此选项替代了不提倡使用的选项 --memaccess）。
- AAPCS 不再要求位域缺省为无符号。因此，增加了一个新的选项 --signed_bitfields，用于指定位域是有符号的。缺省为 --unsigned_bitfields。
- 以前，ARM 编译器为目的地址未知的任何符号生成 .directive 节。此要求已被 ELF 标准机制取代。这表示使用 dllimport 导入的符号的行为可能有所改变。例如：

  ```
  __declspec(dllimport)  int f();
  int g() { return f(); }
  ```

  不生成任何 .directive 节。
- 编译器提供新的 __user_setup_stackheap() 函数，该函数采用与 __user_initial_stackheap() 相同的方式返回初始堆栈和堆的位置。__user_setup_stackheap() 减小了代码大小 （优于 __user_initial_stackheap()），因为不需要临时堆栈。

- ARM 编译器现在支持命令行选项 --diag_style gnu，以显示与 gcc 报告的格式相匹配的消息。这是对早期版本中使用的 arm 和 ide 格式的增补。
- 现在，嵌入式汇编器在每个函数开始时的状态由编译器的调用来设置，并由 #pragma arm 和 #pragma thumb 编译指示来修饰。

  通过在嵌入式汇编器函数中使用显式 ARM、THUMB 或 CODE16 指令，您还可以在函数中更改嵌入式汇编器的状态，仅限当前函数。

A.2.3 RVCT v3.0 中库支持的改动

RVCT v3.0 中有以下改动：

- 《ARM 体系结构的 C 库 ABI》[CLIBABI] 描述了由编译器检查的一致性测试 _AEABI_PORTABILITY_LEVEL。使用它可以提高您的代码到 CLIBABI 的其他实现的可移植性。
- 最新版本的 *《ARM 体系结构的 ABI （基本标准）》*[BSABI] 纠正了 assert() 宏的定义中的一个缺陷，并要求遵守信号一致性。现在，它定义为 __aeabi_assert()。有关详细信息，请参阅 assert.h。
- RVCT v3.0 为线程安全的字符串函数 strlcpy 和 strlcat 提供了库支持。
- RVCT v3.0 包括 __user_initial_stackheap() 的新的库实现。这表示如果您使用包含 ARM_LIB_STACK、ARM_LIB_HEAP 和/或 ARM_LIB_STACKHEAP 指令的分散加载文件，则不需要重新实现此函数。
- RVCT v3.0 中有了新的库机制，在这种机制中，当代码引用诸如 __use_no_heap 或 __use_no_semihosting 等 __use_no_* 符号时，如果检测到符号冲突，错误消息可以提供更多信息。
- 为了完全符合 POSIX， armar 不再接受不带 - 前缀的命令行选项。这表示如下形式的命令：

  ```
  armar cru lib.a foo.o
  ```

  是错误的。如果您使用早期版本的 RVCT 的编译脚本或 makefile，则这些命令可能需要更改。
- armar 现在支持命令行选项 --diag_style gnu，以显示与 gcc 报告的格式相匹配的消息。这是对早期版本中使用的 arm 和 ide 格式的增补。

A.2.4 RVCT v3.0 中 ARM 链接器的改动

RVCT v3.0 中有以下改动：

- 在当前版本中，链接器仅支持 SVr4 映像和共享库的线程局部存储 (TLS)。有关链接器实现的完整信息，请参阅 ARM 体系结构的 ABI 附录和勘误表 [ABI 附录]。
- RVCT v3.0 包括新的链接器选项 `--compress_debug`，它强制压缩 `.debug_*` 节，以便消除一些冗余以及减少调试表大小。但使用 `--compress_debug` 选项将导致链接时间延长。
- RVCT v3.0 包括新的链接器选项 `--info libraries`，它可以输出为链接阶段自动选择的每个库的完整路径名称。可以将此选项与修饰符 `--info_lib_prefix` 一起使用，以显示有关特定库的信息。
- RVCTv3.0 包含新的链接器选项 `--no_legacyalign`，在放置节时强制链接器使用自然对齐。
- RVCT v3.0 包括新的链接器选项 `--dynamic_debug`，强制链接器输出调试节的动态重定位。
- RVCT v3.0 包括新的链接器选项 `--show_cmdline`，查看链接器如何处理命令行。
- ARM 链接器现在支持命令行选项 `--diag_style gnu`，以显示与 `gcc` 报告的格式相匹配的消息。这是对早期版本中使用的 `arm` 和 `ide` 格式的增补。

A.2.5 RVCT v3.0 中 ARM 汇编器的改动

RVCT v3.0 中有以下改动：

- RVCT v3.0 支持 ARMv7 的 Thumb-2 执行环境 （Thumb-2EE）体系结构扩展。此项扩展基于 Thumb-2。关键区别是：
  - 新的 `ENTERX` 和 `LEAVEX` 状态更改指令，可用在 Thumb 状态和新的 ThumbEE 状态中
  - 新的 `HB`、`HBL`、`HBLP` 和 `HBP` 指令，用于跳转至处理程序
  - 空指针，用于检查加载和存储
  - ThumbEE 状态下额外的 `CHKA` 指令，用于检查数组边界
  - 对加载、存储和跳转指令 （`BX`、`BLX` 和 `BXJ`）的某些其他修改。

- RVCT v3.0 包括对 ARMv7 体系结构的两个组件的初步支持，它们是高级 SIMD 扩展和 VFPv3。

  高级 SIMD 扩展包括：

  — 高级 SIMD 寄存器。它们与 VFPv3 寄存器组相同，不过被看作 32 个 64 位寄存器或 16 个 128 位寄存器。

  — 将高级 SIMD 寄存器作为元素的向量来处理的大量新指令。

  这些新指令可用在 ARM 和 Thumb-2 指令集中。

  VFPv3 与 VFPv2 相比有以下增强功能：

  — 将 VFPv2 寄存器组大幅扩展至 32 个 64 位寄存器，是以前的 VFP 寄存器组的一倍。这些寄存器不包括主要的 ARM 通用寄存器。

  — 多个新的指令可以高效地加载大量的常用浮点数，以及高效地转换浮点和定点格式。

  这些新指令可用在 ARM 和 Thumb-2 指令集中。

- 作为 ARM 对高级 SIMD 指令的支持的一部分，*浮点状态和控制寄存器* (FPSCR) 的第 27 位现在被保留为累积标记，以便指示饱和整数运算时是否发生饱和。

- ARM 汇编器可用于汇编 Intel 无线 MMX 技术指令，以便为 PXA270 处理器开发代码。引入了两个新指令，加载/存储指令的语法更改为支持符号和文字。

- 使用以下选项将输出以列表形式写入文件：

  `--list` *`file`*

  如果未指定 *`file`*，则使用 `--list=` 将输出发送到 *`inputfile`*`.lst`。

  注意

  您可以使用 `--list` 将输出发送到 `.lst` 文件。不过，不提倡使用此语法，汇编器将发出警告。

- 引入了两个新指令：
  — `RELOC`，在对象文件中对 ELF 重定位进行编码
  — `QN`，定义指定 NEON™ 四字寄存器的名称。

- ARM 汇编器现在支持命令行选项 `--diag_style gnu`，以显示与 `gcc` 报告的格式相匹配的消息。这是对早期版本中使用的 `arm` 和 `ide` 格式的增补。

- 使用 `armasm --debug` 时不再保留局部符号。如果要保留局部符号来辅助调试，必须指定 `--keep`。

- 在 RVCT v2.2 中，在 Thumb 汇编器代码中将 LDR 伪指令与局部标签一起使用会生成一个没有设置 Thumb 位的地址。如果希望您的代码具有此行为，则使用 armasm --untyped_local_labels。

A.2.6 RVCT v3.0 中 fromelf 实用程序的改动

RVCT v3.0 中有以下改动：

- Fromelf 实用程序现在支持 --debugonly 选项，用于删除 ELF 输出文件中的代码和数据节。
- 在 RVCT v3.0 中，fromelf --text 的语法有所不同。以下形式不提倡使用，将在以后的版本中删除：
  - fromelf --text=*xx*
  - fromelf --text/*xx*
- RVCT v3.0 包括新的 fromelf 选项 --show_cmdline，用于查看 fromelf 实用程序如何处理命令行。
- fromelf 实用程序现在支持命令行选项 --diag_style gnu，以显示与 gcc 报告的格式相匹配的消息。这是对早期版本中使用的 arm 和 ide 格式的增补。

A.3 RVCT v2.2 SP1 和 RVCT v2.2 之间的差异

本节介绍随 RealView Developer Suite v2.2 Service Pack 1 (SP1) 提供的 RVCT v2.2 与早期版本之间的差异。

差异如下：

- RVCT v2.2 SP1 包括对 ARM1136J(F)-S-rev-1 内核的支持。要查看支持的内核的完整列表，请使用：
  ```
  armcc --cpu list
  ```
- 在编译共享对象时， ARM 链接器提供新的选项，即使用：
  - `--pt_arm_exidx` 来创建一个描述异常表的位置的 `PT_ARM_EXIDX` 程序头文件。链接器假定包含 `PT_ARM_EXIDX` 程序头文件的共享对象可能引发异常。
  - `--force_so_throw`，强制链接器假定所有共享对象可能引发异常。

  有关这些命令行选项的详细信息，请参阅 《链接器和实用程序指南》中描述 System V 共享库的一章。
- 支持预处理指令 `#warning`。此指令在编译时生成用户定义的警告，但不会暂停编译。
- 支持关键字 **`__restrict`** 和 **`__restrict__`**，作为 **`restrict`** 的同义词。在此版本中，两个关键字可以在所有模式中使用 （而不仅是 `--restrict`）。
- `deprecated` 函数属性在 GNU 模式和 ARM 模式中支持：
  ```
  int Function_Attributes_deprecated_0(int b) __attribute__ ((deprecated));
  ```
- fromelf 实用程序现在支持 `--no_comment_section` 选项，用于删除 ELF 输出文件中的 `.comment` 节。
- 编译器选项 `--enum_is_int` 的存储类型选择已更改。如果枚举器的范围大于 **`signed int`** 的范围，但在 **`unsigned int`** 的范围内，则 **`enum`** 的存储类型现在是 **`unsigned int`**。

  例如：
  ```
  enum E { k = 0x8000000 };              /* is treated as unsigned int */
  ```

- 作为 ARM 汇编语言开发的一部分，SWI 指令已重命名为 SVC（超级用户调用）。

  指令被反汇编为 SVC，并提供注释指明这是以前的 SWI。例如，fromelf --text -c 生成：

```
...
0x00000fbc:    e3a00010    ....    MOV      r0,#0x10
0x00000fc0:    ef123456    V4..    SVC      0x123456 ; formerly SWI
...
```

A.4 RVCT v2.2 和 RVCT 2.1 版之间的差异

本节介绍 RVCT v2.2 与 RVCT v2.1 之间的差异。

A.4.1 RVCT v2.2 中的一般性改动

RVCT v2.2 中有以下改动。

- RVCT v2.2 支持新的 Thumb®-2 指令集。Thumb-2 引入了许多新的 32 位指令，以及一些新的 16 位指令。在支持它的处理器上（如 ARM1156T2(F)-S）， Thumb-2 可提供与 ARM 接近的性能以及与原始 Thumb 类似的代码大小。

  Thumb-2 指令集包括旧的 16 位 Thumb 指令，作为子集。

- RVCT v2.2 支持新的 ARMv6 内核，例如 ARM1176JZ(F)-S （采用了经 ARM TrustZone™ 技术优化的软件）、 ARM968EJ-S、 ARM1156T2(F)-S 和 ARM MPCore™。

  RVCT v2.2 汇编器提供对 MPCore 指令的支持。

- RVCT v2.2 支持新的汇编器语法，同时继续支持旧的语法，以便可以汇编遗留代码。

- RVCT v2.2 与 《ARM 《体系结构的基本平台 ABI》[BPABI] 完全兼容，因此可以支持一系列的操作系统，例如 ARM Linux 和 Symbian OS。

- 按照 *《ARM 体系结构的 ABI（基本标准）》*[BSABI] 的规定，RVCT v2.2 初步支持 DWARF 3 （标准草案 9）调试表。现在提供一个新的命令行选项 `--dwarf3`，用于在编译代码时指定此格式。

- 命令行选项 `--debug` 或 `-g` 可以指定是否为当前编译生成调试表。优化选项由 `-O`*num* 指定。缺省情况下，使用 `--debug` 或 `-g` 选项不会影响优化设置。

  这是 RVCT v2.2 中的行为改动 （单独使用 `--debug` 或 `-g` 相当于在 RVCT v2.1 中使用 `[--debug|-g] -O0`）。除此新的缺省行为外，优化选项 （即 `-O`*num*、`-Ospace` 或 `-Otime`）的行为没有改动。

- RVCT v2.2 支持命令行选项 `--apcs /fpic`，用于编译与 System V 共享库兼容的代码。使用此选项可以生成只读的、与位置无关的代码，其中相对地址引用与程序被加载的位置无关。

- ARM 链接器支持生成共享库以及链接到共享库。提供新的命令行选项来生成 SVr4 可执行文件和共享对象，以及指定生成代码的方式。

- ARM 链接器符合 《ARM 体系结构的基本平台 ABI》 [BPABI]，以及支持经 GNU 扩展的符号版本模型。
- 浮点数计算的 ARM 实现已更改，改进了对 C99 函数的支持。在此更改显著改变行为的情形中，引入了*兼容模式*，以辅助开发人员迁移代码以使用新的功能。有关详细信息，请参阅第A-26 页的*RVCT v2.2 中库支持的改动*。
- ARM 编译器 C 实现已更改，现在对超出范围的枚举器值发出警告。这些值的处理方式与在 C++ 中相同。这表示当您升级到最新版本的 RVCT 时，具有超范围值的 `enum` 类型的大小可能与 C 中的大小不同。

  如果您使用 C++ 或命令行选项 `--enum_is_int`、`--strict` 或 `--strict_warnings`，则没有变化。有关详细信息，请参阅第A-24 页的*RVCT v2.2 中 ARM 编译器的改动*。
- ARM 库已增强，改进了对多线程的支持。它的目的是为了帮助使用基于 RTOS 的系统的开发人员。
- ARM 编译器提供新的选项，以更好地控制动态符号的导出方式：
  - `--dllexport_all`
  - `--no_hide_all`
- ARM 链接器可以执行 RVCT 的其他组件不可用的某些跳转优化。提供两个新的命令行选项来处理尾调用节，在节的尾部优化跳转指令。
- RVCT v2.1 中不提倡使用的选项在 v2.2 中现已不再使用 （请参阅*RVCT v2.2 中不再使用的功能*）。
- RVCT v2.1 中支持的某些选项在 v2.2 中已不提倡使用 （请参阅第A-24 页的*RVCT v2.2 中不提倡使用的功能*）。

RVCT v2.2 中不再使用的功能

RVCT v2.2 中有以下改动：

- RVCT v2.1 中不提倡使用的所有功能和选项在 RVCT v2.2 中已不再使用，这些功能和选项有：
  - 传统的*软件开发工具包* (SDT) 格式 （如 *ARM 对象格式* (AOF) 和） *ARM 库格式* (ALF) 格式的库。
  - 对关键字使用单短线，例如 `armlink -help`。
  - 编译器选项 `-ansi` 和 `-ansic`。

- RVCT v2.1 中不提倡使用的旧编译器选项名称在 RVCT v2.2 中已不再使用，例如 -fy、-fd、-Ec 和 -zo。

  当存在优先使用的选项名称时，编译器会发出警告，例如：

  ```
  armcc -zo
  Warning: X0010W: Old syntax, please use '--split_sections'.
  ```

  要查看这些功能和选项的列表，请参阅 《编译器用户指南》中的 “附录 A：使用旧命令行选项”。

- 不再使用某些旧的 ARM 处理器和体系结构：
  - ARM6
  - ARMv3 和 ARMv3M

  要查看支持的内核的完整列表，请使用：

  ```
  armcc --cpu list
  ```

- 不再使用某些旧的浮点体系结构：
  - VFPv1 （缺省为 VFPv2）
  - FPA
  - 软件 FPA

  除非另行指定，否则缺省为软件 VFP。

  要查看支持的浮点体系结构的完整列表，请使用：

  ```
  armcc --fpu list
  ```

- 以下编译器选项不再使用：
  - --fpu fpa、--fpu softfpa 和 --fpu vfpv1
  - --fa
  - --cpu 3 和 --cpu 3M
  - -Oldrd 和 -Ono_ldrd
  - -W*letter* 和 -E*letter*

- 不支持 SXT 和 UXT 指令 （符号扩展或零扩展）的同义词。
- 不再使用 FPA 寄存器 f0-f7 和 F0-F7。
- 不支持使用环境变量 RVCT21_CLWARN 来对不提倡的选项发出警告。

RVCT v2.2 中不提倡使用的功能

RVCT v2.2 中有以下改动：

- 编译器支持选项 `--apcs /adsabi` 来编译与旧的 *ARM Developer Suite*™ (ADS) *应用程序二进制接口* (ABI) 兼容的代码。此选项不提倡使用，将在以后的版本中删除。
- ARM 链接器和 `fromelf` 接受两种形式的否定选项，例如 `--no_debug` 和 `--nodebug`。但不提倡使用非标准形式，并且以后将不再支持非标准形式，例如 `--nodebug`。如果您使用不提倡使用的语法，则将发出警告。
- C++ 配置选项 `--dll_vtbl` 已由新的 `--export_all_vtbl` 选项取代。选项 `--dll_vtbl` 不提倡使用，以后将不再支持。
- RVCT 汇编器支持两种形式的独占加载寄存器指令：
  - `LDREX{B|D|H}{`*cond*`} `*Rd*`, [`*Rn*`]`
  - `LDR{B|D|H}EX{`*cond*`} `*Rd*`, [`*Rn*`]`

  第二种形式不提倡使用，将在以后的版本中删除。

  反汇编器仅支持第一种形式。
- RVCT 汇编器支持两种形式的独占存储寄存器指令：
  - `STREX{B|D|H}{`*cond*`} `*Rd*`, [`*Rn*`]`
  - `STR{B|D|H}EX{`*cond*`} `*Rd*`, [`*Rn*`]`

  第二种形式不提倡使用，将在以后的版本中删除。

  反汇编器仅支持第一种形式。

A.4.2 RVCT v2.2 中 ARM 编译器的改动

RVCT v2.2 中有以下改动：

- ARM 编译器提供新的选项，以更好地控制动态符号的导出方式，即使用：
  - `--export_all_vtbl` 来导出具有关键函数的类的所有虚拟表函数和 RTTI
  - `--export_defs_implicitly` 来导出原型被标记为 `dllimport` 的定义。
- 当生成共享对象或 DLL 时， ARM 编译器提供新的选项来指定符号可见性，即使用：
  - `--dllexport_all` 来提供所有全局符号的动态可见度，除非另行指定
  - `--no_hide_all` 来导出所有 `extern` 定义以及导入所有未定义的引用。

- 新的 __swi_indirect_r7 的行为与 __swi_indirect_r12 相类似，只不过它使用 r7 而不使用 r12。ARM Linux 上的 Thumb 应用程序使用 __swi_indirect_r7 进行内核系统调用。
- --force_new_nothrow 的行为已扩充，它可以使任何重载的全局 operator new 被作为 throw() 来处理。
- ARM 编译器 C 实现已更改，现在对超出范围的枚举器值发出警告。

  在严格的 C 中，枚举器值必须可以表示为 **int**，例如它们的范围必须为 -2147483648 至 +2147483647（含这两个值）。在早期版本的 RVCT 中，超出范围的值将转换为 **int**，而不会发出警告 （除非指定了 --strict 选项）。

  在 RVCT v2.2 中，这些值的处理方式与在 C++ 中相同，即将它们当作 **unsigned int**、**long long** 或 **unsigned long long** 处理。这表示如果您使用最新版本的 RVCT，则具有超范围值的 **enum** 类型的大小可能与在 C 中的大小不同。例如：

```
enum E1 { k1 = 0xffffffff };  /* value == 2147483648u; out-of-range in C */
    /* C: before 2.2: sizeof(enum E1) == 1 */
    /* C:        2.2: sizeof(enum E1) == 4 */
    /* C++:      all: sizeof(enum E1) == 4 */
enum E2 { k2 = (int)0xffffffff };  /* value == -1; in range */
    /* C: before 2.2: sizeof(enum E1) == 1 */
    /* C:        2.2: sizeof(enum E1) == 1 */
    /* C++:      all: sizeof(enum E1) == 1 */
enum E3 { k3 = -1, k4 = 0xffffffff };  /* value == 2147483648u; out-of-range in C */
    /* C: before 2.2: sizeof(enum E1) == 1 */
    /* C:        2.2: sizeof(enum E1) == 8; use long long */
    /* C++:      all: sizeof(enum E1) == 8; use long long */
```

  要确保报告超出范围的警告，请使用以下命令将它们更改为错误：

  ```
  armcc --diag_error 66 ...
  ```

  如果您使用 C++ 或命令行选项 --enum_is_int、--strict 或 --strict_warnings，则没有变化。

- ARM 编译器现在支持在 ARM 和 GNU 模式中使用 __attribute__ 关键字。
- ARM 编译器现在支持新的变量属性 zero_init 或 __zero_init__，用于指定不具有初始值设定项的变量被放置在 ZI 数据节中。
- ARM 编译器提供新的选项 --dwarf3，用于指定 DWARF 3 标准调试表，以便当编译以 C 或 C++ 编写的代码时描述 ARM 或 Thumb 程序。如果您不指定格式，则编译器假定为 DWARF 2。
- ARM 编译器支持新的环境变量 RVCT22_CCOPT，它可用于指定编译器的命令行选项。

- RVCT v2.2 支持命令行选项 --apcs /fpic，用于编译与 System V 共享库兼容的代码。使用此选项可以生成只读的、与位置无关的代码，其中相对地址引用与程序被加载的位置无关。
- ARM 编译器提供新的内在函数来控制中断处理：
  - __enable_irq() 和 __disable_irq()
  - __enable_fiq() 和 __disable_fiq()
- ARM 编译器提供新的内在函数来控制优化：
  - __schedule_barrier()
  - __force_stores()
  - __memory_changed()
- ARM 编译器包括新的选项，用于启用或禁用根据上下文隐式确认模板参数从属名称是有类型还是无类型，即使用：
  - --implicit_typename
  - --no_implicit_typename

  缺省为 --no_implicit_typename。
- 当调用编译器时，您可以使用 --show_cmdline 选项来查看编译器如何处理命令行。命令以标准化方式显示，并展开所有 via 文件的内容。但使用此选项不会捕获命令行中的意外错误。
- 对于优化级别 -O3 的浮点算法，ARM 编译器不再保证符合 ISO C 和 C++ 标准。您必须使用 --fpmode=std 选项来确保符合 ISO C 和 C++ 标准。
- 按照 ISO C 标准的规定， ARM 编译器现在将 **const volatile** （和 **volatile const**）数据放置在 RW 或 ZI 节中。以前，这些数据被放置在 RO 节中。如果您错误地假定了编译器放置数据的方式，则可能影响到您的代码。

A.4.3 RVCT v2.2 中库支持的改动

RVCT v2.2 中有以下改动：

- 浮点数计算的 ARM 实现已更改，改进了对 C99 函数的支持。在此更改显著改变行为的情形中，引入了*兼容模式*，以辅助开发人员迁移代码以使用新的功能。fplib 中新增的 （或行为有所改动的） C99 函数有：
  - ilogb、 ilogbf、 ilogbl
  - logb、 logbf、 logbl
  - scalbn、 scalbnf、 scalbnl、 scalbln、 scalblnf、 scalblnl

— nextafter、nextafterf、nextafterl、nexttoward、nexttowardf、nexttowardl

mathlib 中新增的（或行为有所改动的）C99 函数有：

— fpclassify 和 signbit

— isfinite、isinf、isnan 和 isnormal

— copysign、copysignf

— isgreater、isgreaterequal、isless、islessequal、islessgreater 和 isunordered

为了帮助您移植代码，新的兼容模式可以模拟这些函数和宏以前的行为：

— ilogb、ilogbf、ilogbl

— finite

— isnan

注意

在将来的版本中将删除此旧式支持。ARM 建议您将使用的这些功能改为最新版本编译器中的等效函数。

- ARM 库已增强，改进了对多线程的支持。它的目的是为了帮助使用基于 RTOS 的系统的开发人员。

  用户可覆盖的函数 __user_libspace() 已拆分为两个包装函数：

  __user_perproc_libspace()

  它返回指向 __user_libspace 数据区的指针，该数据区用于存储整个进程的全局数据，即在所有线程中共享的数据。

  __user_perthread_libspace()

  它返回指向 __user_libspace 数据区的指针，该数据区用于存储特定进程的局部数据。

  还有三个新的用户可覆盖函数来管理锁机制，防止同时访问造成共享数据损坏：

  _mutex_initialize()

  此函数接受指向 32 位字的指针，并将它作为有效的互斥量来初始化。

  ```
  int _mutex_initialize(mutex *m);
  ```

  _mutex_acquire()

  此函数使调用线程在提供的互斥量上获得锁。

  ```
  void _mutex_acquire(mutex *m);
  ```

_mutex_release()

此函数使调用线程释放提供的互斥量。

```
void _mutex_release(mutex *m);
```

这导致某些函数在多线程环境中的行为方式有所改变，更方便开发人员在多进程系统中使用这些函数。

- ARM 库管理程序 armar 支持命令行选项 --diag_style，以指定诊断消息的显示方式。例如要包含行号和字符计数，可使用：

  ```
  armar --diag_style ide
  ```

  缺省为 ARM 格式，即 --diag_style arm。

A.4.4 RVCT v2.2 中 ARM 链接器的改动

RVCT v2.2 中有以下改动：

- ARM 链接器符合 《ARM 体系结构的基本平台 ABI》 [BPABI]，并支持经 GNU 扩展的符号版本模型。链接器提供新的选项来控制符号版本，即使用：
  - — --symver_script *file* 来启用隐式符号版本，以及将 *file* 作为符号版本脚本输入。
  - — --symver_soname 来启用隐式符号版本和版本符号，以便强制静态绑定。

    如果符号没有已定义的版本，链接器将使用所链接的文件的 SONAME。

  注意

  通常情况下，符号版本只在生成或链接到 DSO 或共享库时有用。它对静态链接没有影响。

- ARM 链接器可以执行 RVCT 的其他组件不可用的某些跳转优化。

  两个新的命令行选项可以控制这些优化，即使用：

  - — --inline 来启用跳转内联。缺省情况下，内联为禁用。
  - — --info inline 在每内联一个函数时显示信息，以及查看内联的总数。

  当您指定 --inline 和 --feedback *file* 时，由链接器内联的函数也被发布到反馈文件中。

- 提供两个新的命令行选项来处理尾调用节，在节的尾部优化跳转指令，即使用：
  - `--tailreorder` 将尾调用节移到其目标之上 （如果可能）
  - `--info tailreorder` 来显示有关已移动的尾调用节的信息。
- 两个新命令行选项可用于在 ARM 链接器中控制中间代码生成，即使用：
  - `--no_inlineveneer` 来禁用内联中间代码
  - `--no_veneershare` 来防止中间代码共享。
- ARM 链接器支持生成共享库以及链接到共享库。提供新的命令行选项来编译 SVr4 和 BPABI 可执行文件和共享对象 BPABI DLL，以及指定生成代码的方式，即使用：
  - `--sysv` 来编译 SVr4 格式的 ELF 文件
  - `--shared` 来编译 SVr4 共享对象
  - `--soname` *name* 来指定共享对象的 SONAME
  - `--fpic` 来链接与位置无关的代码
  - `--init` *symbol* 来指定初始化代码
  - `--fini` *symbol* 在卸载可执行文件或共享对象时执行代码
  - `--linux_abitag` *id* 来指定最低兼容 Linux 内核版本
  - `--dynamiclinker` *name* 来更改缺省动态链接器。
- 提供新的链接器选项 `--startup` 来将备选的 C 库与不同的启动符一起使用。类似地，`--cppinit` 选项可用于 C++ 初始化代码。
- ARM 链接器选项 `--symbols` 列出链接步骤中使用的局部和全局符号。在 RVCT v2.2 中，缺省情况下，此输出不再列出映射符号。新的命令行选项 `--list_mapping_symbols` 现在可以将映射符号包括在 `--symbols` 生成的输出中。
- 如果链接器检测到指定 ARMv3 （在 RVCT v2.2 中已不再使用）的对象，它会将这些对象升级到 ARMv4 以便可以和 ARM 库一起使用。当链接器提升目标体系结构级别时，将显示警告消息。
- ARM 链接器使您可以在分散加载描述文件中按符号名称来引用输入节。请参阅 《链接器和实用程序指南》中对 *input_symbol_pattern* 的描述。

A.4.5 RVCT v2.2 中 ARM 汇编器的改动

RVCT v2.2 中有以下改动：

- ARMv6T2 定义了 Thumb-2，它与 Thumb 指令集相比有了重大改进。Thumb-2 提供了几乎与 ARM 指令集完全相同的功能。它同时具有 16 位和 32 位指令，并可同时实现类似于 ARM 的性能以及类似于 Thumb 的代码密度。

  ARMv6T2 还定义了 ARM 指令集中的多个新指令。

  汇编器支持 ARM 和 Thumb-2 中的所有新指令。

- RVCT v2.2 包括对新的 ARMv6 体系结构扩展的支持：
  - ARMv6Z 定义 ARM 安全扩展 (TrustZone)，例如用于 ARM1176JZ(F)-S 内核。
  - ARMv6K 定义*对称的多处理器*系统 (SMP) 的指令，例如用于 ARM MPCore。

- RVCT v2.2 汇编器可用于编写可以汇编为 ARM 或 Thumb-2 指令的源代码。您可以使用相同的语言来为 Thumb-2 之前的处理器编写 Thumb 指令。

  但 RVCT v2.2 也支持旧的汇编语言语法，以便能够汇编遗留代码。

- ARM 汇编器支持 `COMMON` 指令，用于在指定的符号处分配所定义大小的内存块。您还可以指定内存的对齐方式：

  `COMMON` *`symbol`*`{,`*`size`*`{,`*`alignment`*`}}`

- ARM 汇编器提供一个新选项 `--dwarf3`，用于指定 DWARF 3 标准调试表。DWARF 2 仍然是缺省值。

- ARM 汇编器提供了新的选项，用于在生成共享对象或 DLL 时指定符号可见性，即使用：
  - `--dllexport_all` 来提供所有全局符号的动态可见度，除非另行指定
  - `--no_hide_all` 来导出所有 `extern` 定义以及导入所有未定义的引用。

- 通过对 `IMPORT` 和 `EXPORT` 指令使用新的属性， ARM 汇编器可以输出具有可见度集的 ELF 符号：
  - `DYNAMIC`
  - `HIDDEN`
  - `PROTECTED`

A.4.6 RVCT v2.2 中对 fromelf 实用程序的改动

RVCT v2.2 中有以下改动：

- fromelf 实用程序的功能已增强，可以在 `--text=/s` 的输出中支持经 GNU 扩展的符号版本表。使用新的 `-- no_symbolversions` 选项可以禁用符号版本表的解码。

A.5 RVCT v2.1 与 RVCT v2.0 之间的差异

本节介绍 RVCT v2.1 与 RVCT v2.0 之间的差异。

A.5.1 RVCT 2.1 版中的一般性改动

RVCT 2.1 版中有以下改动：

- 完全支持 C++，包括异常。
- 增强了对 ARMv6 内核的支持。要查看支持的内核的完整列表，请使用：
  ```
  armcc --cpu list
  ```
- 编译器、链接器、汇编器和 `fromelf` 支持新的 `--diag_style` 选项，以生成与 IDE （如 Microsoft Visual Studio）更兼容的格式的警告和错误。
- 编译器和链接器支持新的实用程序，以删除生成的 C++ 代码中未使用的虚拟函数。
- 编译器支持新的 `--min_array_alignment` 选项。
- 提供链接器反馈信息，以便在下次编译文件时通知编译器有未使用的函数。这些函数将放置在各自的节中，以便链接器将来删除它们。
- 按照《ARM 体系结构的 ELF》标准 [AAELF] 的定义， ARM® 工具可以使用 `SHF_STRINGS` 节在多个编译单元中共享合适的字符串。
- 不提倡使用 VFPv1。新的缺省为 VFPv2。要查看支持的 FPU 的完整列表，请使用：
  ```
  armcc --fpu list
  ```
- 不提倡对关键字使用单短线，以后将不再支持。使用编译工具时使用双短线。
- 缺省情况下，使用不提倡的选项 （如编译器选项 `--fpu softfpa`）时编译工具将发出警告。

  在 RVCT v2.1 中，可以通过将环境变量 `RVCT21_CLWARN` 设置为以下值之一来更改此行为：

  0 对使用旧语法和不提倡选项的情况发出警告。

  1 接受旧语法，不对其发出警告，但在使用不提倡选项时发出警告。这是缺省设置。

  2 接受旧语法和不提倡使用选项，不发出警告。

A.5.2 RVCT 2.1 版中 ARM 编译器的改动

RVCT 2.1 版中有以下改动：

- 当使用 --gnu 选项运行编译器时，支持 GNU 扩展。但在不使用此选项运行编译器时也支持一些扩展。这些编译模式被称为：

  ARM 模式

  缺省模式，即不使用 --gnu 选项进行编译。

  GNU 模式

  使用 --gnu 选项进行编译。

  有关所有 GNU 扩展以及支持的模式和语言的完整列表，请参阅 *《RealView 编译工具 2.1 版编译器和库指南》*中描述编译器引用的一章。

- 通过 Edison Design Group (EDG) 前端增强了对 ISO C++ 的支持。它提供完整的 C++ 解析程序，将程序表示形式传递到 ARM 编译器，以生成代码。它现在支持引发和捕获 C++ 异常。

- 多文件编译提供在多个编译单元中实现优化。使用新的 --multifile 选项可以指定此行为。多文件编译需要在命令行上指定多个文件，例如：

  ```
  armcc [options] --multifile ifile_1 ... ifile_n
  ```

- 新的 -O3 优化级别，缺省情况下包括多文件编译。

- 新的 --cpu list 和 --fpu list 选项可以显示有关支持的 CPU 和体系结构的详细信息。

- 新的 --min_array_alignment 选项可用于指定阵列的最低对齐要求。

- 新的 __breakpoint() 内在函数。

- Noreturn 函数。

- --old_cfe 选项现在已不再使用。

A.5.3 RVCT 2.1 版中库支持的改动

RVCT 2.1 版中有以下改动：

- C++ 库（即 Rogue Wave 和 C++ 运行时库）现在支持 C++ 异常。C++ 库继续支持不需要异常支持的应用程序。

- C 库现在支持所有 wchar.h 函数 （文件 I/O 除外），以及 printf 和 scanf 中的 c99 十六进制浮点支持。

- 引入了新的区表格式来支持压缩算法。此新格式不再包含 ZISection$$Table。

A.5.4 RVCT 2.1 版中 ARM 链接器的改动

RVCT 2.1 版中有以下改动：

- 缺省情况下，启用读/写数据压缩以优化 ROM 大小。
- 提供新的选项 --rosplit 来输出两个 RO 执行区，一个用于代码，另一个用于数据。
- 提供新的命令行选项来支持 C++ 异常表。使用新的选项 --noexceptions 来确保您的代码中没有任何异常。

  新的选项 --exceptions_tables=unwind|nounwind 强制链接器生成异常表，而与输入文件的内容无关。例如，使用 --exceptions_tables=unwind，链接器可以为具有调试帧信息的 C 和汇编语言对象创建异常表。
- 提供新的选项 --userlibpath 来指定搜索用户库的位置。
- 现在，链接器在检查对象文件的对齐时更加严格。它确保要求堆栈 8 字节对齐的任何代码仅由保留堆栈 8 字节对齐的代码直接或间接地调用。如果检测到堆栈对齐冲突，链接器将生成错误消息：

  Error L6238E: *object_name.o(section_name)* contains invalid call from ' ~PRES8' function to ' REQ8' *function_name*

  如果引用了外部符号地址，也会生成类似的警告消息：

  Warning L6306W: ' ~PRES8' section *object_name.o(section_name)* should not use the address of ' REQ8' *function_name*

A.5.5 RVCT 2.1 版中 ARM 汇编器的改动

RVCT 2.1 版中有以下改动：

- 新的 --cpu list 和 --fpu list 选项可以显示有关支持的 CPU 和体系结构的详细信息。
- 汇编器包括新的 :RCONST: unary 运算符，用于返回给定寄存器的编号。
- 汇编器检查修饰堆栈指针 (SP) 的指令，以确定是否将代码标记为 PRES8。如果需要，可以自动执行此更改（请参阅 *《RealView 编译工具 2.1 版汇编器指南》*中描述指令引用的一章）。

- 汇编器可以发出有关代码中可能存在互锁的警告。要启用此选项，请使用：

  ```
  armasm --diag_warning 1563
  ```

A.5.6 RVCT 2.1 版中对 fromelf 实用程序的改动

RVCT 2.1 版中有以下改动：

- 新的 `--expandarrays` 选项可以解码 ELF 映像，以便在结构的内部和外部展开阵列。

  此选项只能与 `--text -a` 选项一起使用。

A.6 RVCT v2.0 与 RVCT v1.2 之间的差异

本节介绍 RVCT v2.0 与 RVCT v1.2 之间的差异。

A.6.1 RVCT v2.0 中的一般性改动

RVCT v2.0 中有以下改动：

- 支持 ARM 体系结构 v6。
- 符合 *ARM 体系结构的 ABI（基本标准）*[BSABI]。有关详细信息，请参阅 `http://www.arm.com`。
- 要引发浮点异常，必须选择 `--fpmode ieee_full`。这是因为缺省设置现在是 `--fpmode std`，因此缺省情况下不生成浮点异常。
- 支持使用双短线 “`--`”指示命令行关键字 （例如 `--cpp`）。

A.6.2 RVCT 2.0 版中 ARM 编译器的改动

ARM 编译器 (`armcc`) 有以下更改：

- RVCT 2.0 版编译器有新的前端，包括对命令行选项做了更改。为了向后兼容，也支持老版本 ARM 编译器中提供的选项。
- 现在，四个独立的编译器 （`armcc`、`tcc`、`armcpp` 和 `tcpp`）已经合并为一个编译器 `armcc`。但为了有助于移植到新的编译器，可使用单独的编译器名称来调用 RVCT 2.0 版编译器。
- 支持 ARMv6 及其未对齐访问特性。
- 添加新的嵌入式汇编器与内联编译器配合使用。
- 使用 `#pragma arm` 和 `#pragma thumb` 实现每个函数的 ARM 和 Thumb® 编译。
- 使用 `--fpmode` 选项的五个浮点模型。
- `--list` 选项的行为与在旧的编译器中的行为不同。
- C++ 模板实例化。
- C++ 命名空间。
- 您可以指定指针对齐的级别。
- 编译 C++ 时不再支持 ROPI。

- 诊断消息的控制和处理。同样，诊断消息的编号方式已经更改。现在，消息的编号格式为 #nnnn 或 #nnnn-D。带有 -D 后缀的消息的消息编号可用于使您能够处理诊断消息的选项中。
- 新接口不支持许多旧版本的编译器选项。但为了向后兼容，如果使用 --old_cfe 选项，则可以使用这些选项。有关详细信息，请参阅 *《RealView 编译工具 2.0 版编译器和库指南》*中介绍旧版本编译器选项的附录。适用时，此附录也说明旧版本编译器选项映射到新版本编译器选项的方式。对于在 *《RealView 编译工具 2.0 版编译器和库指南》*中列出的消息，该附录也说明新编译器接口所输出的等效消息。

  注意

  如果使用 --old_cfe 选项，编译器所输出的消息将使用旧的编号格式。

其他更改包括添加了新的编译指示和预定义宏、其他 C 和 C++ 语言扩展以及对 ARM C 和 C++ 库进行了更改。

A.6.3 RVCT 2.0 版中 ARM 链接器的改动

ARM 链接器 (armlink) 有以下更改：

- --unresolved 选项现在可应用于部分链接。
- 添加了新的控制文件命令 RESOLVE，并在执行部分链接时使用。RESOLVE 使用时与 armlink 选项 --unresolved 相似。
- 选项 --edit 现在可以接受多个文件。
- 有了指定填充字节值的新选项 --pad。
- 添加了新的分散加载属性 EMPTY 和 ZEROPAD。

A.6.4 RVCT 2.0 版中 ARM 汇编器的改动

ARM 汇编器 (armasm) 有以下更改：

- 添加了对新的 ARM 体系结构 v6 的支持。包括饱和指令、并行指令和组合及分离指令。
- ALIGN 指令具有附加参数，可指定任何填充的内容。此参数为可选。
- 具有新的 AREA 指令 NOALLOC。
- 具有两个新指令， ELIF 和 FRAME RETURN ADDRESS。
- 具有四个新的内置变量 {AREANAME}、 {COMMANDLINE}、 {LINENUM} 和 {INPUTFILE}。