RealView® Development Suite

Version 3.1

Getting Started Guide

ARM®

ARM DUI 0255G

RealView Development Suite
Getting Started Guide

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
September 2003	A	Non-Confidential	RVDS v2.0 Release
January 2004	B	Non-Confidential	RVDS v2.1 Release
December 2004	C	Non-Confidential	RVDS v2.2 Release
May 2005	D	Non-Confidential	RVDS v2.2 SP1 Release
March 2006	E	Non-Confidential	RVDS v3.0 Release
March 2007	F	Non-Confidential	RVDS v3.1 Release
February 2008	G	Non-Confidential	Release for RVDS v3.1 Professional

Proprietary Notice

Confidentiality Status

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
RealView Development Suite Getting Started Guide

Preface
About this book viii
Feedback xii

Chapter 1 Introduction
1.1 RealView Development Suite components 1-2
1.2 RealView Development Suite licensing 1-7
1.3 RealView Development Suite documentation 1-8
1.4 RealView Development Suite examples 1-9
1.5 RealView Profiler examples (Professional only) 1-11
1.6 Debug Interface support 1-12
1.7 Fixing problems with your RVDS environment 1-13

Chapter 2 Changes to RealView Development Suite
2.1 Changes between RVDS v3.1 Professional and RVDS v3.1 2-2

Chapter 3 Getting Started with RealView Development Suite
3.1 Building and debugging task overview 3-2
3.2 Using the example projects 3-5
3.3 Getting started with RealView Profiler (Professional only) 3-6

Appendix A **Using the armenv Tool**
A.1 About the armenv tool A-2
A.2 Using the armenv tool A-3

Appendix B **About Previous Releases**
B.1 Changes between RVDS v3.1 and RVDS v3.0 ..... B-2
B.2 Changes between RVDS v3.0 SP1 and RVDS v3.0 ..... B-5
B.3 Changes between RVDS v3.0 and RVDS v2.2 SP1 ..... B-6
B.4 Changes between RVDS v2.2 SP1 and RVDS v2.2 ..... B-10
B.5 Changes between RVDS v2.2 and RVDS v2.1 ..... B-11
B.6 Changes between RVDS v2.1 and RVDS v2.0 ..... B-13
B.7 Changes between RVDS v2.2 and ADS v1.2.1 ..... B-15

Preface

This preface introduces the *RealView® Development Suite Getting Started Guide*, that shows you how to start using RealView Development Suite to manage software projects and to debug your application programs. It contains the following sections:

- *About this book* on page viii
- *Feedback* on page xii.

About this book

RealView Development Suite (RVDS) provides tools for building, debugging, and managing software development projects targeting ARM® architecture-based processors. This book contains:

- an introduction to the software components that make up RVDS
- a summary of the differences between RVDS v3.1 Professional and previous RVDS releases
- a list of the main RVDS environment variables.

Intended audience

This book has been written for developers who are using RVDS to manage development projects for ARM architecture-based processors. It assumes that you are an experienced software developer, but might not be familiar with the ARM development tools.

Using this book

This book is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to RVDS components, licensing, and documentation.

Chapter 2 ***Changes to RealView Development Suite***

Read this chapter for a description of the changes between RVDS v3.1 Professional and the previous release.

Chapter 3 ***Getting Started with RealView Development Suite***

Read this chapter for an overview of the main tasks that you can do with the RVDS tools. It also describes the example projects provided with RVDS.

Appendix A ***Using the armenv Tool***

Read this appendix for a description of how to use the `armenv` tool.

Appendix B ***About Previous Releases***

Read this chapter for a description of previous RVDS releases.

Typographical conventions

The following typographical conventions are used in this book:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`monospace italic`	Denotes arguments to commands and functions where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
`...`	At the end of a path name `...` denotes that the directories of interest are below the last-specified directory name. The unspecified path names are usually those that are different between operating systems. For example: *`install_directory`*`\ARM\RVDS\Examples\...` In the middle of a path name `...` denotes that additional directories exist between the directory names specified. The unspecified path names are usually version and build numbers and platform-specific directory names. For example: *`install_directory`*`\ARM\RVD\Core\...\etc`

Further reading

This section lists publications from both ARM Limited and third parties that provide additional information.

ARM Limited periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata, addenda, and Frequently Asked Questions (FAQs).

ARM publications

See the following documentation for details of the FLEX*net* license management system, supplied by GLOBEtrotter Inc., that controls the use of ARM applications:

- *ARM FLEXnet License Management Guide v4.1* (ARM DUI 0209).

Make sure that you use version 4.1 of this document for details on license management in RVDS v3.1.

This book is part of the RVDS documentation suite. Other books in this suite include:

- *RealView Profiler User Guide* (ARM DUI 0412)
- *RealView Development Suite Eclipse Plug-in User Guide* (ARM DUI 0330)
- *RealView Development Suite CodeWarrior IDE Guide* (ARM DUI 0065)
- *RealView Development Suite Glossary* (ARM DUI 0324)
- *RealView Compilation Tools Essentials Guide* (ARM DUI 0202)
- *RealView Compilation Tools Developer Guide* (ARM DUI 0203)
- *RealView Compilation Tools Assembler Guide* (ARM DUI 0204)
- *RealView Compilation Tools Compiler User Guide* (ARM DUI 0205)
- *RealView Compilation Tools Compiler Reference Guide* (ARM DUI 0348)
- *RealView Compilation Tools Libraries and Floating Point Support Guide* (ARM DUI 0349)
- *RealView Compilation Tools Linker and Utilities Guide* (ARM DUI 0206)
- *RealView Compilation Tools NEON™ Vectorizing Compiler Guide* (ARM DUI 0350)
- *RealView Debugger Essentials Guide* (ARM DUI 0181)
- *RealView Debugger User Guide* (ARM DUI 0153)
- *RealView Debugger Target Configuration Guide* (ARM DUI 0182)
- *RealView Debugger Trace User Guide* (ARM DUI 0322)
- *RealView Debugger RTOS Guide* (ARM DUI 0323)
- *RealView Debugger Command Line Reference Guide* (ARM DUI 0175)
- *RealView ARMulator ISS v1.4 User Guide* (ARM DUI 0207).

The following documentation provides general information on the ARM architecture, processors, associated devices, and software interfaces:

- *ARM Reference Peripheral Specification* (ARM DDI 0062)
- the ARM datasheet or technical reference manual for your hardware device.

For general information on software interfaces and standards supported by ARM, see *install_directory*\Documentation\Specifications\.

Refer to the following documentation for information relating to the ARM debug interfaces suitable for use with RealView Development Suite:

- *RealView ICE and RealView Trace User Guide* (ARM DUI 0155).

Other publications

For a comprehensive introduction to ARM architecture see:

Steve Furber, *ARM system-on-chip architecture* (2nd edition, 2000). Addison Wesley, ISBN 0-201-67519-6.

For more information about CEVA-Oak, CEVA-TeakLite, and CEVA-Teak processors from CEVA, Inc. see http://www.ceva-dsp.com.

Feedback

ARM Limited welcomes feedback on both RVDS and its documentation.

Feedback on RealView Development Suite

If you have any problems with RVDS, contact your supplier. To help them provide a rapid and useful response, give:

- your name and company
- the serial number of the product
- details of the release you are using
- details of the platform you are running on, such as the hardware platform, operating system type and version
- a small standalone sample of code that reproduces the problem
- a clear explanation of what you expected to happen, and what actually happened
- the commands you used, including any command-line options
- sample output illustrating the problem
- the version string of the tool, including the version number and date.

Note

If you have any problems with RealView Debugger, it is recommended that you create a Software Problem Report. To do this, select **Send a Problem Report...** from the RealView Debugger **Help** menu. See the RealView Debugger online help for more details.

Feedback on this book

If you have any comments on this book, send email to `errata@arm.com` giving:

- the document title
- the document number
- the page number(s) to which your comments apply
- a concise explanation of your comments.

General suggestions for additions and improvements are also welcome.

Chapter 1
Introduction

This chapter introduces *RealView® Development Suite* (RVDS) v3.1. It describes the component applications, the additional licenses you can purchase to extend the features of RVDS v3.1, and gives an overview of the documentation suite.

This chapter contains the following sections:

- *RealView Development Suite components* on page 1-2
- *RealView Development Suite licensing* on page 1-7
- *RealView Development Suite documentation* on page 1-8
- *RealView Development Suite examples* on page 1-9
- *RealView Profiler examples (Professional only)* on page 1-11
- *Debug Interface support* on page 1-12
- *Fixing problems with your RVDS environment* on page 1-13.

1.1 RealView Development Suite components

RVDS provides a coordinated development environment for embedded systems applications running on the ARM® family of RISC processors. It consists of a suite of tools, together with supporting documentation and examples. The tools enable you to write, build, and debug your applications, either on target hardware or software simulators.

1.1.1 RVDS installation, examples, and documentation directories

Various RVDS directories that are installed on your system contain useful files. The RVDS documentation refers to these directories where required.

All directories can be found below the main installation directory. Also, many of the examples used in the documentation are contained in a single examples directory. Any exceptions are identified where they are referenced.

The main installation, examples, and documentation directories are identified in Table 1-1. The *install_directory* shown is the default installation directory. If you specified a different installation directory, then the path names are relative to your chosen directory.

Table 1-1 RealView Development Suite directories

Directory	Windows default path	Red Hat Linux default path
install_directory	C:\Program Files\ARM	~/arm
Examples	*install_directory*\RVDS\Examples\...	*install_directory*/RVDS/Examples/...
RealView Profiler examples	*install_directory*\RVP\...\examples\...	*install_directory*/RVP/.../examples/...
Project templates	*install_directory*\project_templates\...	*install_directory*/project_templates/...
Documentation	*install_directory*\Documentation\...	*install_directory*/Documentation/...

For more information on accessing the documentation, see *RealView Development Suite documentation* on page 1-8.

For a summary of the examples provided, see:

- *RealView Development Suite examples* on page 1-9
- *RealView Profiler examples (Professional only)* on page 1-11.

1.1.2 Eclipse Plug-in for RVDS

The Eclipse Plug-in for RVDS integrates the RealView development tools into the Eclipse IDE. It enables you to use the Eclipse IDE as a project manager to create, build, and manage C and C++ projects for ARM targets. The Eclipse Plug-in provides ARM project types to simplify the creation of ARM projects, and provides comprehensive configuration panels to specify options for the ARM compiler, assembler, linker, debugger, and other related tools.

For more details, see the *RealView Development Suite Eclipse Plug-in User Guide*.

1.1.3 RealView Compilation Tools

You can use the RealView Compilation Tools to build programs from C, C++, or ARM assembly language source. RealView Compilation Tools comprises the following tools:

- ARM and Thumb C and C++ compiler, `armcc`
- NEON™ Vectorizing compiler is invoked using the command `armcc --vectorize`, see *RealView Development Suite licensing* on page 1-7
- ARM and Thumb assembler, `armasm`
- ARM linker, `armlink`
- ARM librarian, `armar`
- ARM image conversion utility, `fromelf`
- supporting libraries.

For more details on the features available in RealView Compilation Tools, see the *RealView Compilation Tools Essentials Guide*.

For a complete description of the RealView Compilation Tools and associated utilities, and how to use them, see the RealView Compilation Tools documentation. The documentation is listed in *RealView Development Suite documentation* on page 1-8. Also, see the ARM web site for updates and patches to the RealView Compilation Tools as they become available.

1.1.4 RealView Debugger

RealView Debugger together with a supported debug target (see *Debug Interface support* on page 1-12), enables you to debug your application programs and have complete control over the flow of the program execution so that you can quickly isolate and correct errors.

Note

For information specific to using RealView Debugger on Red Hat Linux see the appendix that describes RealView Debugger on Red Hat Linux. You can find this appendix in the *RealView Debugger User Guide*.

RealView Debugger includes the support for:

- multiprocessor debugging
- *Digital Signal Processor* (DSP) debugging
- trace, analysis and profiling
- *operating system* (OS) awareness.

The default license for RealView Debugger enables you to debug applications that run on single or multiple ARM architecture-based processors. However, you must purchase additional licenses to extend the RealView Debugger functionality to support debugging on DSPs. See *RealView Development Suite licensing* on page 1-7 for more details.

For more details on the features available in RealView Debugger, see the *RealView Debugger Essentials Guide*.

For a complete description of RealView Debugger and how to use it, see the RealView Debugger documentation. The documentation is listed in *RealView Development Suite documentation* on page 1-8.

RealView Debugger downloads

To access various RealView Debugger downloads, from RealView Debugger select:

Help → ARM on the Web → Goto RTOS Awareness Downloads

This displays the *OS-Aware and Middleware Debug* web page on the ARM web site. From here you can locate and download the OS plug-in of your choice.

Help → ARM on the Web → Goto Update and Utility Downloads

This displays the *ARM Technical Support - Downloads* web page on the ARM web site. From here you can locate and download any ARM software updates and utilities.

1.1.5 RealView Profiler (Professional only)

RealView Profiler enables you to see how your code performs on a target system, either by observing your code on actual target hardware using RealView ICE and RealView Trace 2 or by testing code against an ARM *Real-Time System Model* (RTSM). On completion RealView Profiler produces an analysis file with detailed information about each function and instruction.

Note

RealView Trace 2 is only available for Windows platforms.

See the *RealView Profiler User Guide* for more information.

1.1.6 RealView ICE v3.2 host software (Professional only)

RealView ICE v3.2 host software is installed with the **Full** product selection. However, for hardware profiling you also need the following:

- RealView ICE run control unit connected to the host using TCP/IP or USB
- RealView Trace 2 data capture unit connected to the host using USB.

Note

RealView Trace 2 is only available for Windows platforms.

See the *RealView ICE and RealView Trace User Guide* for more information.

1.1.7 RealView ARMulator Instruction Set Simulator

RealView ARMulator Instruction Set Simulator (RVISS) simulates the instruction sets and architecture of ARM processors, together with a memory system and peripherals.

RVISS enables you to begin developing and debugging your embedded applications without target hardware. This is useful where hardware is still being developed, or if there is a limited number of development boards available.

RVISS interface connections are available on Windows and Red Hat Linux.

Table 1-2 Default RVISS debug configurations

Debug configuration	Simulated target
RVISS	ARM7TDMI
RVISS_1	ARM926EJ-S
RVISS_2	ARM1176JZF-S

See the *RealView ARMulator ISS User Guide* for more information.

1.1.8 Instruction Set System Model

ISSM simulates the instruction sets and architecture of the Cortex™ family of ARM processors. The default ISSM configuration is configured with the Cortex-A8 target.

See the *RealView Debugger Target Configuration Guide* for more information.

1.1.9 Real-Time System Models (Professional only)

The following list of RTSMs are provided with RealView Profiler:

- ARM926 *Emulation Board* (EB)
- ARM1136 EB
- ARM1176 EB
- Cortex-A8 EB.

There are no equivalent default RTSM configurations in RealView Debugger. To use these models in RealView Debugger, you must create ISSM debug configurations.

For more information on ISSM debug configurations, see:

- *RealView Debugger User Guide*
- *RealView Debugger Target Configuration Guide*.

1.2 RealView Development Suite licensing

All licensing for RVDS is controlled by the FLEXnet license management system. Use the FLEXnet server software to track and control your RVDS licenses. You can request licenses using the ARM Web Licensing page at `http://license.arm.com`. See the *ARM FLEXnet License Management Guide* for details.

This section describes the RVDS licenses that are separately available for RealView tools and associated features.

1.2.1 RealView Profiler license

The RealView Profiler license enables you to analyze the performance of your code during run-time. See the *RealView Profiler User Guide* for more information.

Note

A license is provided with RVDS v3.1 Professional.

1.2.2 NEON vectorizing compiler license

The NEON vectorizing compiler license enables you to use the NEON vectorizing compiler that targets ARM processors with a NEON unit, such as Cortex-A8. See the *RealView Compilation Tools NEON Vectorizing Compiler Guide* for more information.

Note

A license is provided with RVDS v3.1 Professional.

1.2.3 CEVA-Oak and CEVA-Teaklite DSP debugging license

The CEVA-Oak and CEVA-Teaklite *Digital Signal Processor* (DSP) support license enables you to debug applications running on CEVA-Oak and CEVA-Teaklite DSPs.

1.2.4 CEVA-Teak DSP debugging license

The CEVA-Teak DSP support license enables you to debug applications running on CEVA-Teak DSP.

1.2.5 StarCore SC1200 DSP debugging license

The StarCore SC1200 DSP support license enables you to debug applications running on the SC1200 DSP.

1.3 RealView Development Suite documentation

The documentation provided with RVDS is listed in *ARM publications* on page x. This section describes how to get additional information online.

Also, see the *Further Reading* sections in each book for related publications from ARM Limited, and from third parties.

1.3.1 Getting more information online

Depending on your installation, the full documentation suite is available online as Eclipse viewer and PDF files:

- If you have installed the Eclipse plug-ins, then to view the RVDS documentation in Eclipse select **Help Contents** from the Eclipse **Help** menu.
- Depending on your installation on Windows, to view the PDF documentation select:

  Start → Programs → ARM → RealView Development Suite v3.1 → RVDS v3.1 Documentation Suite.

The Eclipse viewer and PDF files contain the same information. The documentation is installed in the documentation directory shown in Table 1-1 on page 1-2.

The *RVDS Documentation Suite* that can be accessed from a single PDF, `Collection.pdf`. If you install the full documentation suite, a text search of all the PDF files is possible from this collection.

1.4 RealView Development Suite examples

The code for many of the examples in the RVDS documentation is located in the main examples directory (see *RVDS installation, examples, and documentation directories* on page 1-2).

In addition, the directory contains example code that is not described in the documentation. Read the `readme.txt` in each example directory for more information. The examples are installed in the following subdirectories:

`asm` Some examples of ARM assembly language programming. The examples are used in the *RealView Compilation Tools Assembler Guide*.

`cached_dhry` Examples of routines to initialize cache and TCMs on various ARM processors, built around the Dhrystone example. The supported processors include:

- ARM9xx processors
- ARM11xx processors
- Cortex™-A8
- Cortex-R4.

`Cortex-M1` Examples for the ARM Cortex-M1 processor, that include example scatter files and build scripts.

`Cortex-M3` Examples for the ARM Cortex-M3 processor, that include example scatter files and build scripts.

`cpp` Some basic C++ examples.

`databort` Design documentation and example code for a standard Data Abort handler.

`dcc` Example code that demonstrates how to use the Debug Communications Channel. The example is described in the *RealView Compilation Tools Developer Guide*.

`dhrystone` The Dhrystone Benchmark. The example is used in the RealView Debugger documentation.

`dsp` This example demonstrates the use of the ETSI basic operations provided in `dspfns.h`.

`emb_sw_dev` The example projects referenced in the chapter that describes embedded software development in the *RealView Compilation Tools Developer Guide*. The following subdirectories are included:

- `build`*n*. Batch and make files to build the example projects. See the related `readme.txt` file for a description of each project.
- `dhry` Source files for the Dhrystone benchmarking program. This program provides the code base for the example projects in the individual `build`*n* directories.
- `include` User defined header files.
- `scatter` Scatter files used to build the example projects.
- `source` All other source files required to build the example projects.

`fft_v5te` Fast Fourier Transform code optimized for ARM architecture v5TE (ARMv5TE).

`interwork` Examples that show how to interwork between ARM code and Thumb code. See the chapter that describes interworking ARM and Thumb in the *RealView Compilation Tools Developer Guide* for details.

`mmugen` The source and documentation for the MMUgen utility. This utility can generate MMU pagetable data from a rules file that describes the virtual to physical address translation required.

`picpid` An example of how to write position-independent code.

`sorts` Example code that compares an insertion sort, shell sort, and the quick sort used in the ARM C libraries.

`svc` An example *Supervisor Call* (SVC) handler.

`trace` An example application `trace.c` that is used in the tracing tutorial described in the *RealView Debugger Trace User Guide*. The application:

- simulates a small system that reads a set of input data samples and computes the sample average
- provides a framework for common instruction and data trace scenarios.

The image is scatterloaded as defined in the `trace.scat` file.

`unicode` Example code that enables you to evaluate multibyte character support.

`vfpsupport` Example code for enabling and carrying out VFP operations. Also included are various utility files for configuring the debug system when using VFP, and a PDF of *Application Note 133 Using VFP in RVDS*.

1.5 RealView Profiler examples (Professional only)

The code for the profiling examples is located in the RealView Profiler examples directory (see *RVDS installation, examples, and documentation directories* on page 1-2).

The examples are installed in the following subdirectories:

`doom` Example code that runs Doom.

Note

You must download an external shareware file before you can successfully compile and run this example. See the *RealView Profiler User Guide* for more information.

`fireworks` Example code that produces a simulation of exploding fireworks.

`fft` Example code that runs a fast fourier transform.

1.6 Debug Interface support

The Debug Interfaces supported by RealView Debugger in RVDS are shown in Table 1-3.

Table 1-3 Debug Interfaces supported in RVDS v3.1

Debug Interface	Windows	Red Hat Linux
ISSM to connect to:		
• ISSM targets	Yes	Yes
• RTSM targets.	Yes	Yes
RVISS	Yes	Yes
RealView ICE, which includes support for:	Yes	Yes
• connections with RealView ICE	Yes	Yes
• tracing with RealView Trace.	Yes	No
SoC Designer	Yes	No
ARM Ltd. Direct Connection for connections to a Versatile Platform using the built-in RealView ICE Micro Edition (USB) interface	Yes	Yes

Be aware of the following:

- To use the USB connection to the Versatile Platform, you must perform a custom install and make sure the **Direct USB Debug Connection** option is selected.
- To create SoC Designer connections, you must purchase and install RealView SoC Designer.
- To trace using RealView Trace, you must purchase the RealView Trace product.
- You can use RealMonitor with RealView ICE in RealView Debugger. See the *RealView Debugger Target Configuration Guide* for more details on using RealMonitor with RealView ICE.

1.7 Fixing problems with your RVDS environment

If you are having problems running the component applications in RVDS, then make sure your RVDS environment is correctly configured:

- On Red Hat Linux, run the `RVDS31env.posh` script. This is the preferred method of setting up the RVDS environment on Red Hat Linux. See the *RealView Development Suite Installation Guide* for details of running this script.
- On Windows, you can use the `armenv` utility to modify the RVDS environment after installation. See Appendix A *Using the armenv Tool* for more information. This utility is also available on Red Hat Linux systems.

Note

You cannot use the `armenv` utility on custom installations in this release. If you performed a custom installation on Windows, you must set the environment variables yourself (see *The main RVDS environment variables*). On Red Hat Linux, use the `RVDS31env.posh` script.

1.7.1 The main RVDS environment variables

Table 1-4 shows the main RVDS environment variables that must be set on Windows. Replace ... with the path elements of your installation. Use the preferred methods described in *Fixing problems with your RVDS environment* to set these, if possible. Also, make sure that your `PATH` environment variable includes the locations of the various RVDS component application executables.

Table 1-4 Main RVDS environment variables on Windows

Environment variable	Setting
`ARMROOT`	Your installation directory root (*`install_directory`*). The default is `C:\Program Files\ARM`.
`ARMCONF`	Used to locate the RVISS configuration files: *`install_directory`*`\RDI\armperip\...;`*`install_directory`*`\RVARMulator\MPCore\ARMulator\...\win_32-pentium;`*`install_directory`*`\RVARMulator\v6ARMulator\...\win_32-pentium;`*`install_directory`*`\RVARMulator\ARMulator\...\win_32-pentium`
`ARMDLL`	Used to locate the RVISS DLL files: *`install_directory`*`\RVARMulator\MPCore\ARMulator\...\win_32-pentium;`*`install_directory`*`\RVARMulator\v6ARMulator\...\win_32-pentium;`*`install_directory`*`\RVARMulator\ARMulator\...\win_32-pentium;`*`install_directory`*`\RDI\rdimsvr\...\win_32-pentium`

Table 1-4 Main RVDS environment variables on Windows (continued)

Environment variable	Setting
ARMLMD_LICENSE_FILE	The location of your RVDS license file. See the *ARM FLEXnet License Management Guide* for details of this environment variable.
ISSM_ARM_CORTEXDLL	The location of the ISSM models: *install_directory*\ISSModel\Cortex\...\win_32-pentium
RVCT31BIN	The RealView Compilation Tools program executables: *install_directory*\RVCT\Programs\...\win_32-pentium
RVCT31INC	The ARM compiler include files: *install_directory*\RVCT\Data\...\include\windows
RVCT31LIB	The ARM compiler include files: *install_directory*\RVCT\Data\...\lib
RVD_FLASH_BASE	The location of the Flash files for supported development boards. The default is: *install_directory*\RVD\Flash\...\windows
RVDEBUG_HLPPATH	The RealView Debugger online help files: *install_directory*\Documentation\RVD\...\release\windows\onlinehelp
RVDEBUG_INSTALL	The RealView Debugger executables: *install_directory*\RVD\Core\...\win_32-pentium
RVP_ROOT	The installation directory root for RealView Profiler (Professional only) *install_directory*\RVP\Contents\...\

Chapter 2
Changes to RealView Development Suite

This chapter describes the major changes between *RealView® Development Suite* (RVDS) v3.1 Professional and the previous release, RVDS v3.1.

See *Changes between RVDS v3.1 Professional and RVDS v3.1* on page 2-2 for more information.

2.1 Changes between RVDS v3.1 Professional and RVDS v3.1

This section describes the major changes between RVDS v3.1 Professional and RVDS v3.1.

2.1.1 New features in RVDS v3.1 Professional

The following new features are available in RVDS v3.1 Professional:

- RealView Profiler including examples, documentation and the following Real-Time System Models:
  - ARM926 EB
  - ARM1136 EB
  - ARM1176 EB
  - Cortex-A8 EB.
- RealView ICE v3.2 host software
- A license for the NEON Vectorizing Compiler and RealView Profiler
- Eclipse plug-ins:
  - ARM Flash Programmer
  - ARM Assembler Editor
  - CodeWarrior Importer.

Chapter 3
Getting Started with RealView Development Suite

The component products provided with *RealView® Development Suite* (RVDS) enable you to build and debug one or more images that make up your application. This chapter introduces you to the basic tasks for building and debugging with the RVDS tools. It contains the following sections:

- *Building and debugging task overview* on page 3-2
- *Using the example projects* on page 3-5
- *Getting started with RealView Profiler (Professional only)* on page 3-6

3.1 Building and debugging task overview

Table 3-1 is a high-level procedure showing the main tasks for building and debugging applications with the RVDS tools, and where to find the details.

The tasks referred to in the referenced documentation are not necessarily described in the order presented in Table 3-1. If you are using the RVDS tools for the first time, it is suggested that you work through the tasks in the order described in the referenced documents. The sequence presented in Table 3-1 reflects the order in which the tasks might usually be performed.

Table 3-1 Main building and debugging tasks

Step	Description	Reference
1	Decide what image you want to debug: • If you want to debug an existing image, such as a prebuilt example image, continue at step 9. • If you want to build the image for your project, continue at step 2.	*Using the example projects* on page 3-5
2	Choose the RVDS application you want to use to manage and build your projects: • if you want to use the Eclipse IDE, continue at step 4 • if you want to build from the command line using RealView Compilation Tools, continue at step 3.	
3	If you want to use the RealView Compilation Tools directly, then create makefiles or Windows command files containing the required build commands. Continue at step 9 to load and debug your image in RealView Debugger.	*RealView Compilation Tools Essentials Guide* *RealView Compilation Tools Developer Guide* *RealView Compilation Tools Assembler Guide* *RealView Compilation Tools Compiler User Guide* *RealView Compilation Tools Compiler Reference Guide* *RealView Compilation Tools NEON™ Vectorizing Compiler Guide* *RealView Compilation Tools Libraries and Floating Point Support Guide* *RealView Compilation Tools Linker and Utilities Guide*
4	Start the Eclipse IDE.	*RealView Development Suite Eclipse Plug-in User Guide*

Table 3-1 Main building and debugging tasks (continued)

Step	Description	Reference
5	If an Eclipse project already exists, continue at step 7. Otherwise, create an Eclipse project for your application.	*RealView Development Suite Eclipse Plug-in User Guide*
6	Set up the build configuration settings as required to build the image for your application. Continue at step 8.	*RealView Development Suite Eclipse Plug-in User Guide*
7	Open the existing Eclipse project.	*RealView Development Suite Eclipse Plug-in User Guide*
8	Build the image for the Eclipse project.	*RealView Development Suite Eclipse Plug-in User Guide*
9	Start RealView Debugger.	*RealView Debugger Essentials Guide*
10	Configure your debug target and connections as required.	*RealView Debugger User Guide* *RealView Debugger Target Configuration Guide* *RealView ICE and RealView Trace User Guide*
11	Connect to your debug target.	*RealView Debugger Essentials Guide* *RealView Debugger User Guide*
12	Load the image ready for debugging.	*RealView Debugger Essentials Guide* *RealView Debugger User Guide*
13	Prepare any debugging facilities, such as breakpoints and tracepoints.	*RealView Debugger Essentials Guide* *RealView Debugger User Guide* *RealView Debugger Trace User Guide* *RealView Debugger RTOS Guide*
14	Run the image.	*RealView Debugger Essentials Guide* *RealView Debugger User Guide*
15	Perform the required debugging and monitoring tasks, such as stepping, and displaying contents of variables and memory. If using tracepoints, use the trace analysis facilities of RealView Debugger to analyze the trace output.	*RealView Debugger Essentials Guide* *RealView Debugger User Guide* *RealView Debugger Trace User Guide* *RealView Debugger RTOS Guide*

Table 3-1 Main building and debugging tasks (continued)

Step	Description	Reference
16	What is the result of the debugging session? • If there are problems, continue at step 17. • If there are no problems, rebuild your image for final release.	*RealView Development Suite Eclipse Plug-in User Guide* *RealView Compilation Tools Essentials Guide*
17	Decide how to fix any problems in your source code: • use the Eclipse IDE • use another source editor of your choice.	*RealView Development Suite Eclipse Plug-in User Guide*
18	When you have fixed the problem, then you must rebuild, reload, and debug the image: • if you are using the Eclipse IDE, then return to step 8 • if you are using RealView Compilation Tools directly, then return to step 3.	

3.2 Using the example projects

Although your aim is to build and debug your own application images, the tasks described in the RVDS documentation use some of the example projects provided with RVDS (see *RealView Development Suite examples* on page 1-9).

Until you are familiar with the features of the RVDS components it is suggested that you follow the instructions as described. However, many tasks described in the user documentation require that you modify the files in the examples. Before you do this, make a backup copy of the example project files and directories.

3.3 Getting started with RealView Profiler (Professional only)

RealView Profiler enables you to see how your code performs on a target system, either by observing your code on actual target hardware using RealView ICE and RealView Trace 2 or by testing code against an ARM *Real-Time System Model* (RTSM). On completion RealView Profiler produces an analysis file with detailed information about each function and instruction.

Table 3-2 is a high-level procedure showing the main tasks. The sequence presented in Table 3-2 reflects the order in which the tasks might usually be performed.

Table 3-2 Main profiling tasks

Step	Description	Reference
1	Build the image you want to analyze.	*Building and debugging task overview* on page 3-2
2	Start the Eclipse IDE.	*RealView Development Suite Eclipse Plug-in User Guide*
3	If an Eclipse project already exists, continue at step 5.	
4	Otherwise, create an Eclipse project for your application and add your image file.	*RealView Development Suite Eclipse Plug-in User Guide*
5	Select the image you want to analyze.	*RealView Development Suite Eclipse Plug-in User Guide*
6	Decide what collection method you want to use: • If you want to use hardware or create your own run configuration, continue at step 7. • If you want to use a preconfigured RTSM, continue at step 8.	
7	Configure your target connections within Eclipse.	*RealView Profiler User Guide* *RealView ICE and RealView Trace User Guide* *RealView Development Suite Eclipse Plug-in User Guide*
8	Run the image.	*RealView Profiler User Guide* *RealView Development Suite Eclipse Plug-in User Guide*

Table 3-2 Main profiling tasks (continued)

Step	Description	Reference
9	Perform the required profiling tasks, such as analyzing the summary report, code view, charts, and graphs. If there is no need for optimization, rebuild your image for final release.	*RealView Profiler User Guide* *RealView Development Suite Eclipse Plug-in User Guide* *RealView Compilation Tools Essentials Guide*
10	Optimize your source code: • use the Eclipse IDE • use another source editor of your choice.	*RealView Development Suite Eclipse Plug-in User Guide*
11	Return to step 1.	

Appendix A
Using the armenv Tool

This appendix describes the `armenv` tool that you can use to manage your ARM® RealView® product installations. It includes the following sections:

- *About the armenv tool* on page A-2
- *Using the armenv tool* on page A-3.

A.1 About the armenv tool

The armenv tool enables you to:

- set up and remove the environment variables for ARM RealView products
- check for clashes between the ARM RealView products you have installed
- set up different varieties of the same product.

Note

You cannot use the armenv tool for Custom installations in this release of RVDS.

You can find the armenv tool at:

install_directory/bin/*platform*

A.2 Using the armenv tool

This section describes the syntax of the `armenv` command, and shows some examples of how it can be used.

A.2.1 armenv command syntax

The command syntax of the `armenv` tool is:

```
armenv [-r root] [-u] -p product [[--and] -p product]...
[--user|--system|--proc] [--bat|--sh|--csh|--posh|--exec program [args]]
```

The arguments are described in the following section.

A.2.2 armenv command-line arguments

Table A-1 shows the command-line arguments that are available on all platforms.

Table A-1 Generic armenv arguments

Argument	Description
`--help`	Displays help on the command-line arguments.
`-r root`	The absolute path to the root of the product installation, *`install_directory`*. For example, on Windows the default root is: `C:\Program Files\ARM`
`-p product`	The ARM RealView product. See *Product syntax* on page A-5 for more details.
`--and`	Compute settings for all products before this argument, then do the same for those following it. The settings in the second group override those in the first.

Table A-1 Generic armenv arguments (continued)

Argument	Description
`--proc`	Change the environment for the current process only. You cannot use this argument with `--system` or `--user` on Windows.
`--exec`	Execute a program in the new environment. You cannot use this argument with `--bat` on Windows, or with `--sh`, `--csh`, or `--posh` on Red Hat Linux.
`-u`	Attempts to undo the changes to the environment that were made when setting up the product.

Table A-2 shows the command-line arguments that are specific to Windows systems.

Table A-2 armenv arguments specific to Windows

Argument	Description
`--system`	Update the Windows SYSTEM area of the registry. This is the default.
`--user`	Update the Windows USER area of the registry.
`--bat`	Changes the environment for the current command prompt window. This is the default.

Table A-3 shows the command-line arguments that are specific to Red Hat Linux systems. You can specify only one of these.

Table A-3 armenv arguments specific to Red Hat Linux

Argument	Description
`--csh`	Generate a csh syntax shell script.
`--sh`	Generate a sh syntax shell script.
`--posh`	Generate a portable shell script. This is the default.

Product syntax

The syntax for specifying the product is:

`-p category [name] [version [revision]] [-v name value]...`

where:

category The product identifier, for example, `RVDS`.

name Do not use this argument (`armenv` uses the default name `Contents`).

version The version number of the product, for example, `3.1`. If you do not specify a version, the most recent version of the installed product is used.

revision A specific build number for the product. If you do not specify a build number, the most recent build of the installed product is used.

`-v name value`

Identifies a variant of the same product.

name The type of the variant, for example, `platform`. It is suggested that you use only the `platform` variant.

value The specific variant, for example, `linux-pentium`.

For example, you might have the Red Hat Linux variant of RVDS v3.0 installed.

A.2.3 Examples

The following examples show how to use `armenv`:

- To set up the Red Hat Linux environment variables for the csh shell, and for the most recent build of RVDS v3.1, enter:

  ```
  armenv -r ~/ -p RVDS 3.1 -v platform linux-pentium --csh
  ```

- To check for clashes between RealView Compilation Tools v3.1 and RealView Compilation Tools v3.0, enter:

  ```
  armenv -p RVCT 3.1 -p RVCT 3.0
  ```

- To override the RealView Compilation Tools v3.0 settings with the RealView Compilation Tools v3.1 settings, enter:

  ```
  armenv -p RVCT 3.0 --and -p RVCT 3.1
  ```


Appendix B
About Previous Releases

This chapter summarizes the major differences between the previous releases of *RealView® Development Suite* (RVDS), and also *ARM Developer Suite™* (ADS) v1.2.1. The changes are described in:

- *Changes between RVDS v3.1 and RVDS v3.0* on page B-2
- *Changes between RVDS v3.0 SP1 and RVDS v3.0* on page B-5
- *Changes between RVDS v3.0 and RVDS v2.2 SP1* on page B-6
- *Changes between RVDS v2.2 SP1 and RVDS v2.2* on page B-10
- *Changes between RVDS v2.2 and RVDS v2.1* on page B-11
- *Changes between RVDS v2.1 and RVDS v2.0* on page B-13
- *Changes between RVDS v2.2 and ADS v1.2.1* on page B-15.

B.1 Changes between RVDS v3.1 and RVDS v3.0

This section describes the major changes between RVDS v3.1 and RVDS v3.0.

B.1.1 Processor support

Processor support now includes:

- Cortex™-M1
- Cortex-M3 revision 1
- Cortex-R4
- StarCore SC1200 DSP.

B.1.2 Simulator support

RVDS now provides the following simulator support:

- *Instruction Set System Model* (ISSM) simulates the following additional processors:
  - Cortex-M1
  - Cortex-M3 revision 1, which supports cycle counting
  - Cortex-R4.
- RealView SoC Designer, which provides connections to SoC Designer targets. You must purchase RealView SoC Designer separately.
- *Real-Time System Model* (RTSM), which provides connections to RTSM targets.

Support for these is installed with RealView Debugger.

B.1.3 Project template support

The Eclipse New Project Wizard enables you to create new projects for the RVDS component tools depending on the requirements of your application. These projects can be based on project templates supplied with RVDS.

Additional RealView Debugger and RealView Compilation Tools command line options are provided to support the use of RVDS project templates:

- `--no_project`
- `--project` *`filename`*
- `--reinitialize_workdir`
- `--workdir` *`pathname`*.

Also, preconfigured project templates are provided in the directory:

install_directory\project_templates

These project templates are grouped in the following subdirectories:

ARM RealView Development Boards

These project templates include RealView Debugger configurations that enable you to connect to targets through RealView ICE and ARM® Ltd. Direct Connect interfaces as appropriate.

Bare ARM Cores

These project templates include RealView Debugger configurations that enable you to connect to targets through ISSM and RealView ARMulator ISS (RVISS) interfaces as appropriate.

You can also set the project template and working directory values using the following environment variables:

- RVDS_PROJECT
- RVDS_PROJECT_WORKDIR.

For more details about the command line options, see:

- *RealView Compilation Tools Compiler Reference Guide*
- *RealView Debugger User Guide*.

B.1.4 RealView Compilation Tools

Changes to RealView Compilation Tools v3.1 are described in the *RealView Compilation Tools Essentials Guide*.

B.1.5 RealView Debugger

Changes to RealView Debugger v3.1 are described in the *RealView Debugger Essentials Guide*.

B.1.6 IDE support

Eclipse and the Eclipse Plug-in for RVDS are now installed on all supported platforms as part of the RVDS installation. For details on using the Eclipse Plug-in for RVDS, see the *RealView Development Suite Eclipse Plug-in User Guide*.

B.1.7 Documentation changes

The main changes to the RVDS documentation are as follows:

- All RVDS documentation is now available in Eclipse viewer, HTML file format. The Eclipse viewer enables you to search across all the documentation. Although you can view the documentation in a separate web browser, you cannot search across all the documentation.
- Changes to the RealView Debugger documentation are described in the *RealView Debugger Essentials Guide*.
- Changes to the RealView Compilation Tools documentation are described in the *RealView Compilation Tools Essentials Guide*.

B.1.8 Deprecated features

Some features of RealView Compilation Tools v3.1 are deprecated. See the *RealView Compilation Tools Essentials Guide* for details.

B.1.9 Obsolete features

The following features are obsolete in RVDS v3.1:

- Support for *ARM eXtended Debugger* (AXD) and *ARM Symbolic Debugger* (armsd).
- The ARM Developer Suite™ v1.2.1 CD-ROM is no longer provided.
- Support for the Solaris platform.
- Support for the Red Hat Enterprise Linux v3 platform.
- Dynatext documentation is no longer provided.
- RealView Compilation Tools has obsolete features. See the *RealView Compilation Tools Essentials Guide* for details.
- RealView Debugger has obsolete features. See the *RealView Debugger Essentials Guide* for details.

B.2 Changes between RVDS v3.0 SP1 and RVDS v3.0

RVDS v3.0 Service Pack 1 also provides a consolidation of enhancements made in the RealView Compilation Tools and RealView Debugger since the original RVDS v3.0 release, including:

- preliminary support for Cortex™-R4, including compiler support, debugger support, and a new *Instruction Set System Model* (ISSM) model
- improvements to compilation times and DWARF3 debug data sizes over RVDS v3.0
- SIMD NEON™ assembler now extended to include Programmer's notation
- improved user interface for Debug of a multiprocessor MPCore™ target
- additional Cortex-M3 Examples
- Marvell Feroceon-tuned compiler support.

The RealView Debugger Synchronization Control window has been re-engineered, which also includes the synchronization of various actions. A corresponding `SYNCHACTION` CLI command is provided.

For more details, see the *RealView Development Suite v3.0 SP1 Release Notes*.

B.3 Changes between RVDS v3.0 and RVDS v2.2 SP1

This section describes the changes between RVDS v3.0 and RVDS v2.2 SP1.

B.3.1 New features in RVDS v3.0

The following new features are available in RVDS v3.0:

- Support for the TrustZone® architecture.
- Support for *Thumb®-2 Execution Environment* (Thumb-2EE).
- Support for the ARM Cortex processor family:
  - Cortex-A8
  - Cortex-M3.
- Simulator models for the Cortex-A8 and Cortex-M3 processors are now available. These models are accessible through the new ISSM Target Access in RealView Debugger.

B.3.2 Debugger support

The major changes to RealView Debugger v3.0 are as follows:

- RealView Debugger now runs as a single process. The Target Vehicle Server (TVS) no longer exists as a separate entity.
- The Connection Control window has been re-engineered. See the *RealView Debugger User Guide* for more details.
- The features on the **Synch** tab are now available in a separate Synchronization Control window. See the *RealView Debugger User Guide* for more details.
- The Register pane has been re-engineered. You can now create a user-specific view by copying selected registers to a User tab. See the *RealView Debugger User Guide* for more details.
- The RealView Debugger project manager and related functionality has been removed, so you can no longer create projects and build images within RealView Debugger. However, the source code editing and searching features are still available.

  Note

  To create and build your projects in RVDS v3.0, use CodeWarrior for RVDS (see *CodeWarrior for RVDS changes* on page B-8).

- Simulator support has changed. See *Simulator support* for details.
- RealView Broker (RVBroker) has been re-engineered. Although RealView Debugger still runs RVBroker automatically for local host (RVISS) connections, starting RVBroker for remote simulator connections has changed. You must now specify a username when starting RVBroker on a remote workstation. See the *RealView Debugger Target Configuration Guide* for more details.

  Note

  Support for Multi-ICE® direct connect has been removed in RVDS v3.0.

For more details about the changes to RealView Debugger, see the *RealView Debugger Essentials Guide*.

B.3.3 Compilation Tools support

The major changes to RealView Compilation Tools v3.0 are as follows:

- RealView Compilation Tools v3.0 supports Thumb-2EE.
- The ARM assembler can be used to assemble Intel Wireless MMX Technology instructions to develop code for the PXA270 processor.
- RealView Compilation Tools v3.0 provides full support for DWARF 3 (Draft Standard 9.6) debug tables, as described in the *ABI for the ARM Architecture (base standard)* [BSABI].
- The ARM compiler and linker support *Thread Local Storage* (TLS) to enable programs to use multiple threads.
- The ARM compiler supports improved loop optimization.

For more details about the changes to RealView Compilation Tools, see the *RealView Compilation Tools Essentials Guide*.

B.3.4 Simulator support

RVDS now provides the following simulator support:

- ISSM, which simulates Cortex-A8 and Cortex-M3 processors.
- An MPCore simulated target is now available in RealView ARMulator® ISS (RVISS). However, this does not model multiple processors, so connecting to this model connects only to a single processor.

The RDI ARMulator simulated target is no longer available. Use either:

- the `new_arm` connection on the `localhost` Target Access to connect to simulated ARM® processors using RVISS
- the `ISSM` Target Access to connect to one of the Cortex models.

These are both installed with RealView Debugger.

B.3.5 CodeWarrior for RVDS changes

The major changes to CodeWarrior for RVDS are as follows:

- The External Build Wizard is now supported. This is intended to replace the deprecated makefile importer and the Batch File Runner functionality.
- Support for the `.cc` file extension has been added.
- CodeWarrior now warns you when an unrecognized source file extension (such as `.cmd`) is used.
- Panel settings have been added or removed in line with changes to the compilation tools. See the *RealView Compilation Tools Essentials Guide* for details.

For more details, see the *RealView Development Suite CodeWarrior IDE Guide*.

B.3.6 Documentation changes

Apart from documenting the new features of RVDS, the main changes to the RVDS documentation are with the RealView Debugger documentation. The RealView Debugger documentation has been reorganized as follows:

- The information in the *RealView Debugger Extensions User Guide* is now in the following documents:
  - — the chapter that describes DSP support is now included in the *RealView Debugger User Guide*
  - — the chapter that describes Debugging multiple targets is now included in the *RealView Debugger User Guide*
  - — the chapter that describes Tracing in RealView Debugger is now in the *RealView Debugger Trace User Guide*
  - — the chapter that describes OS support is now in the *RealView Debugger RTOS Guide*.
- The chapter that describes connecting to targets in the *RealView Debugger Target Configuration Guide* is now in the *RealView Debugger User Guide*.

- The *RealView Debugger User Guide* has been restructured to be more task-based.
- The *RealView Debugger Project Management Guide* is not provided, because the RealView Debugger project manager has been removed.

For other detailed changes to the RVDS documentation suite, see:

- *RealView Debugger Essentials Guide*
- *RealView Compilation Tools Essentials Guide*.

B.3.7 Deprecated and removed features

The following features are deprecated or removed in RVDS v3.0:

- Support for *ARM eXtended Debugger* (AXD) and *ARM Symbolic Debugger* (armsd) is deprecated.
- The makefile importer and Batch File Runner functionality in CodeWarrior is deprecated.
- Support for remote RealView Debugger connections through Multi-ICE direct connect has been removed. This means that connections to DSP processors is available only with RealView ICE, which you must purchase separately.
- The RealView Debugger project manager and related functionality has been removed.

For details of other deprecated features, see:

- *RealView Compilation Tools Essentials Guide*
- *RealView Debugger Essentials Guide*.

B.4 Changes between RVDS v2.2 SP1 and RVDS v2.2

This section describes the changes between RVDS v2.2 SP1 and RVDS v2.2.

B.4.1 Documentation changes

The changes to the documentation include:

- The *RealView Developer Suite CodeWarrior IDE Guide* is now included, which describes how to use the ARM® features of CodeWarrior.
- The chapter that described getting started with CodeWarrior has been removed from the *RealView Developer Suite Getting Started Guide*, and incorporated into the *RealView Developer Suite CodeWarrior IDE Guide*.
- Changes to the RealView Debugger documentation for the supported DSPs.

B.4.2 Debugger support

The main difference between the debugging tools in RVDS v2.2 SP1 and RVDS v2.2 is with RealView Debugger, which has support for CEVA-Oak, CEVA-TeakLite, CEVA-Teak, ZSP400, and ZSP500 DSPs.

B.4.3 Compilation Tools support

There are minor changes to the compilation tools between RVDS v2.2 SP1 and RVDS v2.2. See the *RealView Compilation Tools Essentials Guide* for details.

B.5 Changes between RVDS v2.2 and RVDS v2.1

This section describes the changes between RVDS v2.2 and RVDS v2.1.

B.5.1 IDE support

The CodeWarrior IDE is now provided to replace the RealView Debugger IDE. The CodeWarrior IDE in RVDS v2.2 is based on Metrowerks CodeWarrior v5.6.

Note

In RVDS v2.2, CodeWarrior for RVDS is supported on Windows XP and Windows 2000 systems only, and is not supplied for Red Hat Linux.

B.5.2 Debugger tool support

The main differences between the debugging tools in RVDS v2.2 and RVDS v2.1 are with RealView Debugger, which has:

- an improved menu structure
- an improved pane handling mechanism
- improved data navigation with the new Data Navigator pane
- internationalization support
- improved source code coloring
- trace, analysis, and profiling enhancements
- enhanced RTOS support
- support for gcc built images
- additional CLI commands, `PRINTDSM` and `TRACEEXTCOND`.

Also, support for standalone editors and the Vi editing mode has been removed from RealView Debugger.

For a detailed list of changes, see the *RealView Debugger Essentials Guide*.

B.5.3 Compilation Tools support

The main differences between the compilation tools in RVDS v2.2 and RVDS v2.1 are:

- RealView Compilation Tools v2.2 includes support for new ARMv6 cores, for example, the ARM1176JZF-S™, incorporating ARM TrustZone® technology-optimized software, the ARM968EJ-S, the ARM1156T2F-S™, and the ARM MPCore™.

- Available in RealView Compilation Tools v2.2, the new Thumb®-2 instruction set introduces many new 32-bit instructions, and some new 16-bit instructions.

  The Thumb-2 instruction set includes older 16-bit Thumb instructions as a subset.

- RealView Compilation Tools v2.2 is fully compliant with the *Base Platform ABI for the ARM Architecture* [BPABI] (unpublished DRAFT).

- RealView Compilation Tools v2.2 provides initial support for DWARF3 (Draft Standard 9) debug tables, as described in the *ABI for the ARM Architecture (base standard)* [BSABI].

- The command-line option `-g` switches on the generation of debug tables for the current compilation. Optimization options are specified by `-O0`, `-O1`, `-O2`, or `-O3`. By default, using the `-g` option does not affect the optimization setting.

  This is a change in behavior for RealView Compilation Tools v2.2.

- RealView Compilation Tools v2.2 supports the command-line option `--apcs /fpic` to compile code that is compatible with System V shared libraries.

- The ARM linker supports building, and linking against, shared libraries. New command-line options are available to build SVr4 executable files and shared objects, and to specify how code is generated.

- The ARM linker supports the GNU-extended symbol versioning model.

- The ARM implementation of floating-point computations has been changed to provide improved support for C99 functions. Where this changes behavior significantly, a compatibility mode has been introduced to aid developers to migrate code to use the new features.

- RealView Compilation Tools v2.2 supports building of Linux applications and shared libraries.

For a detailed list of changes, see the *RealView Compilation Tools Essentials Guide*.

B.5.4 Agilent Probe support

Agilent Probe support is now available as a custom installation option in RVDS v2.2.

B.6 Changes between RVDS v2.1 and RVDS v2.0

This section describes the changes between RVDS v2.1 and RVDS v2.0.

B.6.1 Debugger tool support

The main differences between the debugging tools in RVDS v2.1 and RVDS v2.0 are:

- *ARM eXtended Debugger* (AXD) is included
- *ARM Symbolic Debugger* (armsd) is included
- RealView Debugger has:
  - trace and profiling enhancements
  - enhanced RTOS support
  - new toolbar buttons and menu changes that mean you now have quick access to commonly used features.

B.6.2 Compilation Tools support

The main differences between the compilation tools in RVDS v2.1 and RVDS v2.0 are:

- Increased compliance with the *Application Binary Interface for the ARM Architecture (Base Standard)* (ABI for the ARM Architecture (Base Standard)). See the ABI for the ARM Architecture page at `http://www.arm.com/`.
- C++ exception handling is supported. Therefore, with the exception of export templates, the remainder of ISO C++ is supported as defined by the *ISO/IEC 14822 :1998 International Standard for C++*.
- More optimization features are included, such as multifile compilation and linker feedback.
- Compression of read/write data areas is provided, to further reduce the image size.
- Some GNU C and C++ extensions are supported.
- Many new command-line options have been added to the build tools.
- The single-dash keyword and some command-line options are deprecated.

Note

The tools now check more strictly the requirement for eight-byte stack alignment. The compiler generates code with `PRESERVE8` and `REQUIRE8`. The linker checks that code that requires eight-byte alignment only calls code that preserves eight-byte alignment.

Therefore, this has implications for your legacy assembler code, object files and libraries. You must check that your existing assembly files, object files, or libraries preserve eight-byte alignment and correct them if required. For more details, see the *RealView Compilation Tools Assembler Guide* and the *RealView Compilation Tools Linker and Utilities Guide* for more details.

B.7 Changes between RVDS v2.2 and ADS v1.2.1

This section describes the changes between RVDS v2.2 and ADS v1.2.1.

B.7.1 CodeWarrior IDE changes

The changes between CodeWarrior for RVDS and CodeWarrior for ADS are:

- CodeWarrior for ADS was based on CodeWarrior v4.2. CodeWarrior for RVDS is now based on Metrowerks CodeWarrior v5.6.
- The CodeWarrior Perl plug-in, MWPerl, which provided support for processing Perl scripts in CodeWarrior v4.2 has been removed in CodeWarrior v5.6. It is no longer supported by Metrowerks.
- The ARM tool-specific configuration panels are tailored to RVDS v2.2.
- The separate ARM compilers are combined into a single compiler in RVDS v2.2, therefore there is only one compiler configuration panel in RVDS v2.2.
- You can now run and debug your image with RealView Debugger, in addition to AXD and armsd.
- You can now concatenate libraries.
- You can now import CodeWarrior for ADS projects into CodeWarrior for RVDS.
- The default ARM stationery in CodeWarrior for RVDS does not include a `DebugRel` build target. However, a `DebugRel` build target is created if you import a CodeWarrior for ADS project, to preserve any settings you might have configured for that build target.
- Unlike the ADS compiler, the RealView Compilation Tools compiler does not generate browser information. This functionality is now provided by the builtin language parser of CodeWarrior.
- Code formatting.
- Code completion, including code completion for C++ template classes.
- Go to next/previous function.
- Word wrap when printing.
- Support for source-relative `#includes`.
- Find inside/outside of comments.
- Improved language parser speed and feedback.

- New editor bindings.
- Ability to show and hide the Code and Data columns in the project window.
- Support for workspaces.

Note

All target connection and debugging features in the CodeWarrior IDE are not available in CodeWarrior for RVDS. You must run one of the ARM debuggers to perform these functions.

B.7.2 Debugger changes

The main differences between the debugging tools in RVDS v2.2 and ADS v1.2.1 are:

- RealView Debugger is the latest ARM debugger, which enables you to perform advanced debugging functions such as:
  - multiprocessor debugging
  - OS-aware debugging
  - extended target visibility
  - trace, analysis, and profiling
  - access to the RealView ICE JTAG control unit over Ethernet and USB.
- AXD is enhanced to be able to debug C and C++ programs built with the new RealView Compilation Tools provided with RVDS v2.2.

B.7.3 Compilation Tools changes

The main differences between the build tools in RVDS v2.2 and ADS v1.2.1 are as follows:

- Compliance with the new ABI for the ARM Architecture (Base Standard). See the ABI for the ARM Architecture page at `http://www.arm.com/`. This is different to the old ADS ABI. Some compatibility is provided with the `--apcs /adsabi` command line option.
- There is full ISO C++ support as defined by the *ISO/IEC 14822 :1998 International Standard for C++*, by way of the EDG (Edison Design Group) front-end. This includes exceptions, namespaces, templates, and intelligent implementation of *Run-Time Type Information* (RTTI), but excludes the export of templates.
- Support for some GNU language extensions.

- ARM and Thumb compilation on a per-function basis.
- Re-engineered inline assembler, and a new embedded assembler that enables you to include out-of-line assembly code.
- Linker feedback to remove unused functions.
- Full support for ARM architecture v6 instructions has been added.
- Read/write data compression enables the optimization of ROM size.
- Removal of unused C++ virtual functions.
- Multifile compilation, which performs optimizations across multiple compilation units.
- You can specify a library search path, to indicate where to search for your user libraries.
- You can separate RO code and data into different execution regions.
- There are new scatter-loading attributes.
- Unicode and multibyte characters are supported.
- Compiler intrinsics are available to access the return address of a function, the current stack pointer value, and the current program counter value. An additional intrinsic enables you to insert the `BKPT` instruction in your C or C++ code.
- You can identify a function that does not return, so that the compiler generates more efficient code.
- The C++ name mangling scheme has changed.
- The *ARM Profiler* (`armprof`) is not provided with RealView Compilation Tools.
- The ARM Applications Library is not provided with RealView Compilation Tools.
- Unlike the ADS compiler, the RealView Compilation Tools compiler does not generate browser information.
- There are changes to the assembler, compiler, and linker command-line options.

  Support for double dashes -- to indicate command-line keywords (for example, `--cpp`) and single dashes - for command-line single-letter options, with or without arguments (for example, `-S`).

Note

The single-dash command-line options used in previous releases of ADS and RealView Compilation Tools are still supported for backwards-compatibility.

- The fromelf option `-ihf` has been removed.

Note

The tools now check more strictly the requirement for eight-byte stack alignment. The compiler generates code with `PRESERVE8` and `REQUIRE8`. The linker checks that code that requires eight-byte alignment only calls code that preserves eight-byte alignment. Therefore, this has implications for your legacy assembler code, object files and libraries. You must check that your existing assembly files, object files, or libraries preserve eight-byte alignment and correct them if required. For more details, see the *RealView Compilation Tools Assembler Guide* and the *RealView Compilation Tools Linker and Utilities Guide* for more details.

B.7.4 ARM simulator changes

RealView ARMulator® ISS is the latest version of the ARM simulator. It supports connections through RealView Connection Broker and RDI. When connecting to the simulator through RealView Connection Broker under RealView Debugger, you can have multiple connections to the simulator. You can connect to the RDI interface of RealView ARMulator ISS using RealView Debugger, AXD v1.3, and armsd.

Note

Although you can install ADS in addition to RealView Development Suite v2.2, you must exercise caution if you use both RealView ARMulator ISS and ADS ARMulator. See the *RealView Development Suite v2.2 Release Notes* for more details.