RealView® Development Suite

Version 3.0

Getting Started Guide

ARM®

ARM DUI 0255E

RealView Development Suite
Getting Started Guide

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Change
September 2003	A	RVDS v2.0 Release
January 2004	B	RVDS v2.1 Release
December 2004	C	RVDS v2.2 Release
May 2005	D	RVDS v2.2 SP1 Release
March 2006	E	RVDS v3.0 Release

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

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

This preface introduces the *RealView® Development Suite v3.0 Getting Started Guide*, that shows you how to start using *RealView Development Suite* (RVDS) to manage software projects and to debug your application programs. It contains the following sections:

About this book

RealView Development Suite 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 RealView Development Suite
- a summary of the differences between RVDS v3.0, previous RVDS versions, and *ARM Developer Suite*™ (ADS) v1.2
- a glossary of terms for users new to RealView Development Suite.

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 v3.0 components, licensing, and documentation.

Chapter 2 *Features of RVDS v3.0*

Read this chapter for a description of the new features in RVDS v3.0.

Chapter 3 *Getting Started with RealView Development Suite*

Read this chapter for an overview of the main tasks that a 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 versions, and ADS v1.2.

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.

ARM publications

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

- *ARM FLEXlm License Management Guide v4.0* (ARM DUI 0209).

Make sure that you use version 4.0 of this document for details on license management in RVDS v3.0.

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

- *RealView Debugger v3.0 Essentials Guide* (ARM DUI 0181)
- *RealView Debugger v3.0 User Guide* (ARM DUI 0153)
- *RealView Debugger v3.0 Target Configuration Guide* (ARM DUI 0182)
- *RealView Debugger v3.0 Trace User Guide* (ARM DUI 0322)
- *RealView Debugger v3.0 RTOS Guide* (ARM DUI 0323)
- *RealView Debugger v3.0 Command Line Reference Guide* (ARM DUI 0175)
- *RealView Compilation Tools v3.0 Essentials Guide* (ARM DUI 0202)
- *RealView Compilation Tools v3.0 Developer Guide* (ARM DUI 0203)
- *RealView Compilation Tools v3.0 Assembler Guide* (ARM DUI 0204)
- *RealView Compilation Tools v3.0 Compiler and Libraries Guide* (ARM DUI 0205)
- *RealView Compilation Tools v3.0 Linker and Utilities Guide* (ARM DUI 0206)
- *RealView ARMulator ISS v1.4 User Guide* (ARM DUI 0207)
- *RealView Development Suite AXD and armsd Debuggers Guide* (ARM DUI 0066).
- *RealView Development Suite CodeWarrior IDE Guide* (ARM DUI 0065).
- *RealView Development Suite Glossary* (ARM DUI 0324)

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)
- *Multi-ICE® User Guide* (ARM DUI 0048)
- *ARM MultiTrace™ User Guide* (ARM DUI 0150)
- *ARM Agilent Debug Interface User Guide* (ARM DUI 0158).

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.

For more information about the ZSP400 and ZSP500 processors from the ZSP division of LSI Logic see http://www.zsp.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, you can create a Software Problem Report using the **Help → Send a Problem Report...** menu. See the RealView Debugger documentation 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) v3.0. It describes the component applications, the additional licenses you can purchase to extend the features of RVDS v3.0, and gives an overview of the documentation suite.

This chapter contains the following sections:

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.

This section includes:

- *RealView Development Suite installation, examples, and documentation directories*
- *CodeWarrior for RVDS* on page 1-3
- *RealView Compilation Tools* on page 1-4
- *RealView Debugger* on page 1-4
- *RealView ARMulator Instruction Set Simulator* on page 1-7
- *Instruction Set System Model* on page 1-8
- *RVDS example projects* on page 1-8
- *ARM eXtended Debugger* on page 1-8
- *ARM Symbolic Debugger* on page 1-8.

1.1.1 RealView Development Suite 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 necessary.

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	Sun Solaris and Red Hat Linux default path
install_directory	C:\Program Files\ARM	~/arm
Examples[a]	*install_directory*\RVDS\Examples\...	*install_directory*/RVDS/Examples/...
Documentation[b]	*install_directory*\Documentation\...	*install_directory*/Documentation/...

a. See *RealView Development Suite examples* on page 1-13 for a summary of the examples provided with RealView Development Suite.
b. See *RealView Development Suite documentation* on page 1-11 for more information on accessing the documentation.

1.1.2 CodeWarrior for RVDS

CodeWarrior for RVDS is based on Metrowerks CodeWarrior IDE version 5.7. In RVDS v3.0, CodeWarrior for RVDS is supported on Windows XP and Windows 2000 systems only. It is not supplied with RVDS on Sun Solaris and Red Hat Linux.

Note

Be aware that you must not use the builtin debugging features of CodeWarrior, such as setting breakpoints. After compiling your application, use the **Project → Debug** menu option to start RealView Debugger, then perform all debugging operations within RealView Debugger.

CodeWarrior for RVDS:

- Provides a graphical user interface for managing your software development projects. You can use the CodeWarrior for RVDS to develop C, C++, and ARM assembly language code targeted at ARM and Thumb® processors. It speeds up your build cycle by providing:
  - comprehensive project management capabilities
  - code navigation routines to help you locate routines quickly.
- Enables you to configure the ARM tools to compile, assemble, and link your project code.
- Enables you to organize source code files, library files, and configuration settings into a *project*. Each project enables you to create and manage multiple configurations of build target settings. For example, you can compile a build target for debugging and a build target for release, and target your code at hardware based on an ARM7TDMI®. Build targets can share files in the same project while using their own settings.
- Provides:
  - a source code editor that provides syntax coloring, and is integrated with the CodeWarrior for RVDS browser
  - a source code browser that keeps a database of symbols defined in your code, and enables you to navigate through your source code quickly and easily

— search and replace capabilities that enable you to use grep-style regular expressions, and perform batch searches through multiple files

— file comparison capabilities that enable you to locate, and optionally merge the differences from one text file to another, and to compare the contents of directories.

For details on how to get started with CodeWarrior for RVDS, see *RealView Development Suite CodeWarrior IDE Guide*.

1.1.3 RealView Compilation Tools

You can use the *RealView Compilation Tools* (RVCT) to build C, C++, or ARM assembly language programs. RVCT comprises the following tools:

- ARM and Thumb C and C++ compiler, `armcc`
- 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 RVCT, see the *RealView Compilation Tools Essentials Guide*.

For a complete description of the RVCT tools and utilities, and how to use them, see the RVCT documentation. The documentation is listed in *RealView Development Suite documentation* on page 1-11. Also, see the ARM web site for updates and patches to the RVCT tools as they become available.

1.1.4 RealView Debugger

RealView Debugger together with a supported debug target (see *Target Access support* on page 1-16), 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 Sun Solaris or Red Hat Linux see the appendix that describes RealView Debugger on Sun Solaris and Red Hat Linux. You can find this appendix in the *RealView Debugger User Guide*.

RealView Debugger includes the support for:

- multiprocessor debugging (see *Multiprocessor debugging* on page 1-5)

- *Digital Signal Processor* (DSP) debugging (see *DSP debugging*)
- trace, analysis and profiling (see *Trace, analysis, and profiling* on page 1-6)
- *operating system* (OS) awareness by downloading vendor-specific plugins (see *OS awareness* on page 1-6).

The default license for RealView Debugger enables you to debug applications that run on a single ARM architecture-based processor. However, you can purchase additional licenses to extend the RealView Debugger functionality to debug applications running on multiple processors, and support debugging on DSP. See *RealView Development Suite licensing* on page 1-10 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 1-11.

Multiprocessor debugging

Multiprocessor debugging enables you to debug software systems running on more than one processor. The processors can be on a single development board, or on multiple development boards. In both cases, RealView Debugger uses a different connection for each processor.

With multiprocessor debugging you can debug mixed core systems and synchronize processor operations.

If you use the same processor on multiple board connections, you might have to create new target descriptions. For more details on creating custom targets, see the *RealView Debugger Target Configuration Guide*.

Multiprocessor debugging requires a separately purchased license. See *RealView Development Suite licensing* on page 1-10 for details.

For more details on multiprocessor debugging, see the chapter that describes debugging multiple targets in the *RealView Debugger User Guide*.

DSP debugging

RealView Debugger provides support for debugging the following DSPs:

- CEVA-Oak, CEVA-Teaklite (revisions B and C), and CEVA-Teak (revisions A and B) DSPs

- CEVA-Teak on the Samsung Scorpio II
- LSI Logic ZSP400 and ZSP500 DSPs.

DSP debugging requires separately purchased licenses. See *RealView Development Suite licensing* on page 1-10 for details.

For more details on DSP support, see the chapter that describes DSP support in the *RealView Debugger User Guide*.

Trace, analysis, and profiling

RealView Debugger provides support for tracing with either trace hardware or a hardware simulator. Trace hardware can be:

- processors with *Embedded Trace Macrocell*™ (ETM)
- on-chip trace buffers such as the ARM Embedded Trace Buffer™ (ETB™)
- a Joint Test Action Group (JTAG) interface unit such as RealView ICE.

Basic, non-ETM, trace support is provided by RealView ARMulator® ISS hardware simulator.

For more details on the trace features available in RealView Debugger, see the chapter that describes tracing in the *RealView Debugger Extensions User Guide*.

OS awareness

OS awareness is an extension that is built into RealView Debugger. You must obtain the plugin for the OS you are using before you can use this RealView Debugger extension. With an OS plugin, RealView Debugger references OS threads and resources (such as queues, mailboxes, and semaphores) in addition to the C or assembler source-level symbolic debug information. See *RealView Debugger downloads* on page 1-7 for details on how to obtain the plugin for your OS.

OS awareness in RealView Debugger is vendor-independent. You can download and use OS plugins from more than one vendor. Therefore, you can develop applications for different OS platforms. If you want to use the OS plugins from more than one vendor in the same debugging session, you might have to create new target descriptions. For more details on creating custom targets, see the *RealView Debugger Target Configuration Guide*.

For more details on OS support, see the chapter that describes OS support in the *RealView Debugger RTOS Guide*. Also, see the ARM web site for more information on OS support as it becomes available.

RealView Debugger downloads

You can access various RealView Debugger downloads from the RealView Debugger **Help** menu:

Help → ARM on the Web → Goto RTOS Awareness Downloads

Displays the *OS Aware Debugger* web page on the ARM web site. From here you can locate and download the OS plugin of your choice.

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

Displays the *RealView Debugger - Updates & Utilities* web page on the ARM web site. From here you can locate and download any software updates and utilities.

1.1.5 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.

Table 1-2 shows the RVISS interface connections that are available from the ARM debuggers on Windows, Sun Solaris, and Red Hat Linux. If you connect to RVISS through RealView Connection Broker, you can connect to multiple instances of RVISS to simulate a multiprocessor system.

Table 1-2 RealView ARMulator ISS connections supported on each platform

ARM Debugger	Remote Debug Interface Connections	RealView Connection Broker Connections
RealView Debugger	Windows [a]	Windows, Sun Solaris and Red Hat Linux
AXD	Windows	Not available
armsd	Windows, Sun Solaris and Red Hat Linux	Not available

a. The Remote Debug Interface connection to RVISS in RealView Debugger is deprecated in this release.

For more details on the features available in RVISS, see the *RealView ARMulator ISS User Guide*.

1.1.6 Instruction Set System Model

Instruction Set System Model (ISSM) simulates the instructions sets and architecture of the Cortex™-A8 and Cortex-M3 processors. For more details, see the *RealView Debugger Target Configuration Guide*.

1.1.7 RVDS example projects

Example projects are provided with RVDS (see *RealView Development Suite examples* on page 1-13). These are located in the directory:

`install_directory\RVDS\Examples\...`

To explore the example projects directory from the Windows **Start** menu, select:

Programs → ARM → ARM RealView Development Suite v3.0 → Examples

1.1.8 ARM eXtended Debugger

ARM eXtended Debugger (AXD) is a single-processor debugger, and is provided for Windows only. AXD together with a supported debug target (see *Target Access support* on page 1-16), enables you to debug your application programs and have control over the flow of the program execution so that you can quickly isolate and correct errors.

Note

AXD is provided for legacy ARM7™ and ARM9™ support, and does not work with the RealView ICE JTAG unit.

For details on how to use AXD, see the *RealView Development Suite AXD and armsd Debuggers Guide*.

1.1.9 ARM Symbolic Debugger

ARM Symbolic Debugger (armsd) is a single-processor debugger that you run from a command-line interface. It is available for Windows, Sun Solaris, and Red Hat Linux.

With armsd you can debug your application programs using the *Remote Debug Interface* (RDI) of RVISS (see *RealView ARMulator Instruction Set Simulator* on page 1-7).

Note

RVISS with armsd supports only ARM7 and ARM9 targets.

For details on how to use armsd, see the *RealView Development Suite AXD and armsd Debuggers Guide*.

1.2 RealView Development Suite licensing

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

Note

You do not have to contact ARM Limited directly to request a license.

The RealView Development Suite licenses that are separately available for RealView Debugger features are described in:

- *Multiprocessor debugging license*
- *CEVA-Oak and CEVA-Teaklite DSP debugging license*
- *CEVA-Teak DSP debugging license*
- *LSI Logic ZSP DSP debugging license*.

1.2.1 Multiprocessor debugging license

The multiprocessor debugging license enables you to debug software systems running on more than one processor (see *Multiprocessor debugging* on page 1-5).

1.2.2 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 (see *DSP debugging* on page 1-5).

1.2.3 CEVA-Teak DSP debugging license

The CEVA-Teak *Digital Signal Processor* (DSP) support license enables you to debug applications running on CEVA-Teak DSP (see *DSP debugging* on page 1-5).

1.2.4 LSI Logic ZSP DSP debugging license

The LSI Logic ZSP *Digital Signal Processor* (DSP) support license enables you to debug applications running on ZSP400 and ZSP500 DSPs (see *DSP debugging* on page 1-5).

1.3 RealView Development Suite documentation

This section describes the documentation provided with RVDS. It contains the following sections:

- *List of documents*
- *Getting more information online*.

1.3.1 List of documents

The RVDS documentation comprises:

- *RealView Development Suite Getting Started Guide* (this document)
- *ARM FLEXlm License Management Guide*
- *RealView Debugger Essentials Guide*
- *RealView Debugger User Guide*
- *RealView Debugger Target Configuration Guide*
- *RealView Debugger Trace User Guide*
- *RealView Debugger RTOS Guide*
- *RealView Debugger Command Line Reference Guide*
- *RealView Compilation Tools Essentials Guide*
- *RealView Compilation Tools Developer Guide*
- *RealView Compilation Tools Assembler Guide*
- *RealView Compilation Tools Compiler and Libraries Guide*
- *RealView Compilation Tools Linker and Utilities Guide*
- *RealView ARMulator ISS User Guide*
- *RealView Development Suite AXD and armsd Debuggers Guide*
- *RealView Development Suite CodeWarrior IDE Guide*
- *RealView Development Suite Glossary*.

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

1.3.2 Getting more information online

Depending on your installation, the full documentation suite is available online as PDF and DynaText for Windows and Sun Solaris, and as PDF for Red Hat Linux. The PDF and DynaText files contain the same information. The documentation is installed in the documentation directory shown in Table 1-1 on page 1-2.

RVDS v3.0 includes a new *PDF 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.

For more details, see:

- *RVDS on Windows*
- *RVDS on Sun Solaris and Red Hat Linux*.

RVDS on Windows

On Windows and Sun Solaris systems, the documentation is also available online as DynaText electronic books. The content of the DynaText manuals is identical to that of the PDF documentation.

Select **Programs → ARM** from the Windows **Start** menu. From here select either:

- **RealView Development Suite v3.0 → RVDS v3.0 Documentation Suite** to view the PDF files
- **DynaText Documentation** to view the DynaText files.

RVDS on Sun Solaris and Red Hat Linux

The full documentation suite is available as PDF and DynaText files for Sun Solaris, and as PDF on Red Hat Linux. If you have set up desktop links, use these to access the required documentation.

Note

DynaText files are not installed on Red Hat Linux.

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 *RealView Development Suite 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, built around the Dhrystone example.

`Cortex-M3` A Hello World example for the ARM Cortex™-M3 processor, that inlcudes an example scatter file and a build script.

`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` A small source file that is required to make full use of the header file `dspfns.h`. This file defines a set of DSP-type primitive operations, and demonstrates how to use the inline assembly feature in the ARM compiler.

`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:

- `buildn` 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 `buildn` directories.
- `include` User defined header files.

`scatter`	Scatter files used to build the example projects.
`source`	All other source files needed to build the example projects.

`fft_v5te`	Fast Fourier Transform code optimized for ARM architecture v5TE (ARMv5TE).
`inline`	Examples that show how to use the inline assembler when compiling ARM C and C++ code. See the chapter that describes mixing C, C++ and assembly language in the *RealView Compilation Tools Developer Guide* for details.
`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.
`unicode`	Example code that enables you to evaluate multibyte character support.
`vfpsupport`	Example code for enabling and carrying out *Vector Floating Point* (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 ARM Developer Suite

RVDS also includes the full version of *ARM Developer Suite*™ (ADS) v1.2.1. ADS v1.2.1 is not installed as part of the RVDS installation. If you have to use ADS v1.2.1, you must install it separately.

1.5.1 Considerations when installing both ADS and RVDS

Be aware of the following when installing both ADS and RVDS:

- Although you can install ADS in addition to RVDS, you must exercise caution if you use both the ADS ARMulator that is installed with ADS and RealView ARMulator ISS. See the *RealView Development Suite Release Notes* for details. However, if you have installed ADS, you can connect to RealView ARMulator ISS using the ADS debuggers (see *RealView ARMulator Instruction Set Simulator* on page 1-7).
- You must use the ARM SuiteSwitcher utility if you want to install both ADS and RVDS on your Windows machine. However, you cannot use the SuiteSwitcher utility to switch between different installed versions of ADS. See the ARM website for a free download of SuiteSwitcher.

1.5.2 Installing the older ADS compilation tools

If you are recommended to use an older version of ADS, these are also provided on the same CD-ROM as ADS v1.2.1.

For Windows, an installer is provided for ADS v1.1 and v1.0.1.

For Sun Solaris and Red Hat Linux, only the compilation tools are provided for ADS v1.1 and v1.0.1. No installer is included for these, and you must copy the files manually. The files are in the `ads_1_1` and `ads_1_0_1` directories on the CD-ROM. You must:

1. Decide what components you need for which platforms, and where you want to copy them.
2. Choose the correct executables from platform-dependent directories.
3. Copy the libraries you require.
4. Make sure that your path includes the location of the executables, and that you set the following environment variables:
   - `ARMINC`, to the directory containing the ARM include files
   - `ARMLIB`, to the directory containing the ARM library files.

1.6 Target Access support

A summary of the Target Access supported by the ARM debuggers in RVDS v3.0 and ADS v1.2.1 are shown in the following sections:

- *RVDS Target Access support on Windows*
- *RVDS Target Access support on Sun Solaris and Red Hat Linux* on page 1-17
- *ADS v1.2.1 target vehicle support* on page 1-17.

1.6.1 RVDS Target Access support on Windows

The Target Access supported by the ARM debuggers in the RVDS on Windows are shown in Table 1-3.

Table 1-3 Target Access supported on Windows

Target Access	RealView Debugger v3.0 or later	AXD v1.3.1	armsd
Agilent Debug Interface (ADI)	Yes	Yes	
Angel debug monitor (Remote_A)	Yes	Yes	Yes
Instruction Set System Model (ISSM)	Yes		
Multi-ICE	Yes	Yes	
MultiTrace™	Yes		
RealMonitor	Yes	Yes	
RealView ARMulator ISS	RealView Simulator Broker (`localhost`) only	RDI only	RDI only
RealView ICE	Yes		
RealView Trace	Yes		
Built-in RealView ICE Micro Edition (USB connection) on the Versatile Platform	Yes		

Be aware of the following:

- To use the USB connection to the Versatile Platform, you must perform a custom install and select the option for RealView ICE Micro Edition USB debug support.
- The RealView Trace software is automatically installed with RealView Debugger.

- You can now use RealMonitor with RealView ICE in RealView Debugger. See the *RealView Debugger Target Configuration Guide* for more details on using RealMonitor with RealView ICE.
- MultiTrace support requires that you install the *ARM Trace Debug Tools* (TDT).
- After installation, you must add the ADI DLL (`gateway.dll`) to the `ARM-A-RR` target list in RealView Debugger. For instructions, see the description of working with RDI targets in the *RealView Debugger Target Configuration Guide*.

1.6.2 RVDS Target Access support on Sun Solaris and Red Hat Linux

The Target Access supported by the ARM debuggers in the RVDS on Sun Solaris and Red Hat Linux are shown in Table 1-4.

Table 1-4 Target vehicles supported on Sun Solaris and Red Hat Linux

Target Access	RealView Debugger v3.0 or later	armsd
ISSM (MxDI)	Yes	
RealView ARMulator ISS	RealView Simulator Broker (`localhost`) only	RDI only
RealMonitor	Yes	
RealView ICE	Yes	

You can now use RealMonitor with RealView ICE in RealView Debugger. See the *RealView Debugger Target Configuration Guide* for more details on using RealMonitor with RealView ICE.

1.6.3 ADS v1.2.1 target vehicle support

The AXD debugger in ADS v1.2.1 supports the following target vehicles:

- RealView ARMulator ISS, through the RDI interface only
- RealMonitor
- Multi-ICE
- MultiTrace, which requires that you install TDT
- Agilent Debug Interface (Gateway)
- Angel debug monitor (Remote_A).

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 Sun Solaris or Red Hat Linux, run either the `RVDS30env.sh` or the `RVDS30env.csh` script depending on your shell. This is the preferred method of setting up the RVDS environment on Sun Solaris or Red Hat Linux. See the *RealView Development Suite Installation Guide* for details of running this script.
- When RVDS is installed on Windows, the installation procedure automatically runs the `armenv` utility, unless you deselected the **Update environment variables in registry** option in the advanced setup options during installation. However, you can use the `armenv` utility to modify the RVDS environment after installation. See Appendix A *Using the armenv Tool* for details on how to use the `armenv` utility. This utility is also available on Sun Solaris and 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 page 1-19). On Sun Solaris or Red Hat Linux, use either the `RVDS30env.sh` or the `RVDS30env.csh` script.

1.7.1 The main RVDS environment variables

Table 1-5 shows the main RVDS environment variables that must be set. Replace ... with the path elements of your installation. Use the preferred methods described in *Fixing problems with your RVDS environment* on page 1-18 to set these, if possible. Also, make sure that your PATH environment variable includes the locations of the various RVDS component application executables.

Table 1-5 Main RVDS environment variables on Windows

Environment variable	Setting
ARMROOT	Your installation directory root (*install_directory*). On Windows, the default is: C:\Program Files\ARM
ARMCONF	Used to locate the RVISS and various RDI target configuration files: *install_directory*\RDI\armperip\...\...;*install_directory*\RVARMulator\v6ARMulator\...\...\win_32-pentium;*install_directory*\RVARMulator\ARMulator\...\...\win_32-pentium
ARMDLL	Used to locate the RVISS and various RDI target DLL files: *install_directory*\RVARMulator\v6ARMulator\...\...\win_32-pentium;*install_directory*\RVARMulator\ARMulator\...\...\win_32-pentium;*install_directory*\ARM\RDI\Targets\Remote_A\...\...\win_32-pentium;*install_directory*\RDI\rdimsvr\...\...\win_32-pentium
ARMLMD_LICENSE_FILE	The location of your RVDS license file. See the *ARM FLEXlm License Management Guide* for details of this environment variable.
RVCT30BIN	The RVCT program executables: *install_directory*\RVCT\Programs\...\...\win_32-pentium
RVCT30INC	The RVCT compiler include files: *install_directory*\RVCT\Data\...\...\include\windows
RVCT30LIB	The RVCT compiler include files: *install_directory*\RVCT\Data\...\...\lib
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

Chapter 2
Features of RVDS v3.0

This chapter describes new features of *RealView® Development Suite* (RVDS) v3.0, and gives a summary of the main changes from RVD v1.8. It includes:

2.1 Changes to RVDS v3.0

The major changes to RVDS v3.0 are described in the following sections:

- *New features in RVDS v3.0*
- *Deprecated and removed features*.

2.1.1 New features in RVDS v3.0

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

- Support for the TrustZone® architecture.
- Support for the *Thumb®-2 Execution Environment* (Thumb-2EE) for ARMv7.
- 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 *Instruction Set System Model* (ISSM) Target Access in RealView Debugger.
- Support for Linux application debugging has been added.

2.1.2 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*.

2.2 RealView Debugger changes

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.
- Support for Linux application debugging has been added. See the *RealView Debugger RTOS 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* on page 1-3).

- Simulator support has changed. See *Simulator support* on page 2-5 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*.

2.3 RealView Compilation Tools changes

The major changes to *RealView Compilation Tools* (RVCT) v3.0 are as follows:

- RVCT v3.0 supports the Thumb-2EE for ARMv7.
- The ARM assembler can be used to assemble Intel Wireless MMX Technology instructions to develop code for the PXA270 processor.
- RVCT 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 RVCT, see the *RealView Compilation Tools Essentials Guide*.

2.4 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.

2.5 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.
- The following entries are not supported on the New dialog box:
  - on the **Project** tab, the `PowerParts Application Wizard` and the `PowerParts Component Wizard` entries
  - on the **File** tab, the `Component Catalog File` and the `PowerParts Palette Wizard` entries
  - on the **Object** tab, the `PowerParts Form Wizard` entry.
- 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*.

2.6 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*.

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:

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	Choose the RVDS application you want to use to manage and build your projects: • if you want to use CodeWarrior for RVDS, continue at step 3 • if you want to build from the command line using RVCT, continue at step 2.	
2	If you want to use the RVCT build 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*
3	Start CodeWarrior for RVDS.	*RealView Development Suite CodeWarrior IDE Guide*
4	If a CodeWarrior for RVDS project already exists, continue at step 6. Otherwise, create a CodeWarrior for RVDS project for your application.	*RealView Development Suite CodeWarrior IDE Guide*
5	Set up the build target settings as required to build the image for your application. Continue at step 7.	*RealView Development Suite CodeWarrior IDE Guide*
6	Open the existing CodeWarrior for RVDS project.	*RealView Development Suite CodeWarrior IDE Guide*
7	Build the image for the CodeWarrior for RVDS project.	*RealView Development Suite CodeWarrior IDE Guide*

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

Step	Description	Reference
8	Decide what image to use: • If you want to use an existing image, such as a prebuilt example image, continue at step 9. • If you want to build a new image, return to step 1.	*Using the example projects* on page 3-5
9	Start RealView Debugger.	*RealView Debugger Essentials Guide*
10	Configure your debug target and connections as required.	*RealView Debugger Target Configuration 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*.
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 CodeWarrior IDE Guide* • *RealView Debugger Essentials Guide* • *RealView Compilation Tools Essentials Guide*.
17	Decide how to fix any problems in your source code:	

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

Step	Description	Reference
	• use CodeWarrior for RVDS • use another source editor of your choice.	• *RealView Development Suite CodeWarrior IDE Guide*
18	When you have fixed the problem, then you must rebuild, reload, and debug the image: • if you are using CodeWarrior for RVDS, then return to step 6 • if you are using RVCT directly, then return to step 2.	

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 1-13).

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.

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. It includes:

- *armenv command syntax*
- *armenv command-line arguments*
- *Examples* on page A-5

A.2.1 armenv command syntax

The command syntax of the `armenv` tool is:

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

The arguments are described in *armenv command-line arguments*.

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 Sun Solaris and 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 Sun Solaris and Red Hat Linux systems. You can specify only one of these.

Table A-3 armenv arguments specific to Sun Solaris and 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 [rev]] [-v name value]...`

where:

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

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

version The version number of the product, for example, `3.0`. If you do not specify a version, the most recent version of the installed product is used.

rev 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, `solaris-sparc`.

For example, you might have both Sun Solaris and Red Hat Linux variants of RVDS v3.0 installed. See *Examples* for an example of how to use this argument.

A.2.3 Examples

To set up the Sun Solaris environment variables for the csh shell, and for the most recent build of RVDS v3.0, enter:

```
armenv -r ~/ -p RVDS 3.0 -v platform solaris-sparc --csh
```

To check for clashes between RVCT v3.0 and RVCT v2.2, enter:

```
armenv -p RVCT 3.0 -p RVCT 2.2
```

To override the RVCT v2.2 settings with the RVCT v3.0 settings, enter:

```
armenv -p RVCT 2.2 --and -p RVCT 2.0
```


Appendix B
About Previous Releases

This chapter summarizes the major differences between the previous releases of *RealView® Developer Suite* (RVDS), that is v2.2 SP1, v2.2, v2.1, v2.0, and *ARM Developer Suite™* (ADS) v1.2.1. The changes are described in:

B.1 Changes between RVDS v2.2 SP1 and RVDS v2.2

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

- *Documentation changes*
- *Debugger support*
- *Build tool support*.

B.1.1 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.1.2 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.1.3 Build tool support

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

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

This section describes the changes between RVDS v2.2 and RVDS v2.1. It contains the following sections:

- *IDE support*
- *Debugger tool support*
- *Build tool support* on page B-4
- *Agilent Probe support* on page B-4.

B.2.1 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 Sun Solaris and Red Hat Linux.

See *CodeWarrior for RVDS* on page 1-3 for more information on CodeWarrior for RVDS.

Note

If you are a user of CodeWarrior for ADS, see *CodeWarrior IDE changes* on page B-7.

B.2.2 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.2.3 Build tool support

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

- RVCT 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 RVCT 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.
- RVCT v2.2 is fully compliant with the *Base Platform ABI for the ARM Architecture* [BPABI] (unpublished DRAFT).
- RVCT 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 `-O`. By default, using the `-g` option does not affect the optimization setting.

  This is a change in behavior for RVCT v2.2.
- RVCT 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.
- RVCT v2.2 supports building of Linux applications and shared libraries.

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

B.2.4 Agilent Probe support

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

B.3 Changes between RVDS v2.1 and RVDS v2.0

This section describes the changes between RVDS v2.1 and RVDS v2.0. It contains the following sections:

- *Debugger tool support*
- *Build tool support*.

B.3.1 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 (see *ARM eXtended Debugger* on page 1-8)
- *ARM Symbolic Debugger* (armsd) is included (see *ARM Symbolic Debugger* on page 1-8)
- 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.3.2 Build tool support

The main differences between the build 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 are now stricter in preserving eight-byte 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.4 Changes between RVDS v2.2 and ADS v1.2.1

This section describes the changes between RVDS v2.2 and ADS v1.2.1. It contains the following sections:

- *CodeWarrior IDE changes*
- *Debugger changes* on page B-8
- *Build tool changes* on page B-8
- *ARM simulator changes* on page B-10.

B.4.1 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 plugin, 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 RVCT 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.4.2 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.4.3 Build tool 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 RVCT.
- The ARM Applications Library is not provided with RVCT.
- Unlike the ADS compiler, the RVCT 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 versions of ADS and RVCT are still supported for backwards-compatibility.

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

Note

The tools are now stricter in preserving eight-byte 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.4.4 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 Developer Suite v2.2 Release Notes* for more details.

