RealView® Developer Suite

Version 2.1

Getting Started Guide

ARM®

ARM DUI 0255B

RealView Developer Suite
Getting Started Guide

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Change
September 2003	A	RVDS Release v2.0
January 2004	B	RVDS Release v2.1

Proprietary Notice

Confidentiality Status

This document is Open Access. This document has no restriction on distribution.

Product Status

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

Web Address

`http://www.arm.com`

Contents
RealView Developer Suite Getting Started Guide

Preface

This preface introduces the *RealView® Developer Suite v2.1 Getting Started Guide*, that shows you how to start using RealView Developer Suite to manage software projects and to debug your application programs. It contains the following sections:

- *About this book* on page vi
- *Feedback* on page ix.

About this book

RealView Developer Suite provides tools for building, debugging, and managing software development projects targeting ARM processors. This book contains:

- an introduction to the software components that make up RealView Developer Suite
- a summary of the differences between RealView Developer Suite v2.1 and *ARM Developer Suite*™ (ADS) v1.2
- a glossary of terms for users new to RealView Developer Suite.

Intended audience

This book has been written for developers who are using RealView Developer Suite 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 RealView Developer Suite v2.1 components, licensing, and documentation.

Chapter 2 ***Differences***

Read this chapter for a description of the differences between RealView Developer Suite v2.1 and ADS v1.2.

Glossary An alphabetically arranged glossary that defines the special terms used.

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.
`monospace`	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.

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 3.1* (ARM DUI 0209).

Make sure that you use version 3.1 of this document for details on license management in RealView Developer Suite v2.1.

This book is part of the RealView Developer Suite documentation suite. Other books in this suite include:

- *RealView Debugger Essentials Guide* (ARM DUI 0181)
- *RealView Debugger User Guide* (ARM DUI 0153)
- *RealView Debugger Target Configuration Guide* (ARM DUI 0182)
- *RealView Debugger Command Line Reference Guide* (ARM DUI 0175)
- *RealView Debugger Extensions User Guide* (ARM DUI 0174)
- *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 and Libraries Guide* (ARM DUI 0205)
- *RealView Compilation Tools Linker and Utilities Guide* (ARM DUI 0206)
- *RealView ARMulator ISS User Guide* (ARM DUI 0207)
- *RealView Developer Suite AXD and armsd Debuggers Guide* (ARM DUI 0066).

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 Developer Suite:

- *RealView ICE 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.

Feedback

ARM Limited welcomes feedback on both RealView Developer Suite and its documentation.

Feedback on RealView Developer Suite

If you have any problems with RealView Developer Suite, 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® Developer Suite v2.1. It describes the component applications, the additional licenses you can purchase to extend the features of RealView Developer Suite v2.1, and gives an overview of the documentation suite.

This chapter contains the following sections:

1.1 RealView Developer Suite v2.1 components

RealView Developer Suite v2.1 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:

1.1.1 RealView Developer Suite installation, examples, and documentation directories

Various RealView Developer Suite directories that are installed on your system contain useful files. The RealView Developer Suite 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 Developer 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 Developer Suite Examples* on page 1-10 for a summary of the examples provided with RealView Developer Suite.
b. See *RealView Developer Suite v2.1 documentation* on page 1-9 for more information on accessing the documentation.

1.1.2 ARM eXtended Debugger

ARM eXtended Debugger (AXD) is a single-processor debugger, and is available for Windows only. AXD together with a supported debug target (see *Target vehicle support* on page 1-13), 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.

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

1.1.3 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 RealView ARMulator® ISS (see *RealView ARMulator Instruction Set Simulator* on page 1-7).

For details on how to use armsd, see the *RealView Developer Suite AXD and armsd Debuggers Guide*. Also, see *RVCT and armsd* on page 1-7.

1.1.4 RealView Debugger

RealView Debugger together with a supported debug target (see *Target vehicle support* on page 1-13), 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.

RealView Debugger also includes the support for:

- multiprocessor debugging (see *Multiprocessor debugging* on page 1-4)
- Digital Signal Processor (DSP) debugging (see *DSP debugging* on page 1-4)
- trace, analysis and profiling (see *Trace, analysis, and profiling* on page 1-5)
- Real-Time Operating System (RTOS) awareness by downloading vendor-specific plugins (see *RTOS awareness* on page 1-5).

You can also build your applications from RealView Debugger because it provides a build interface to the RealView Compilation Tools (see *RealView Compilation Tools* on page 1-6). From this interface you can specify the build options required by your applications. You can also edit your application source code using the built-in text editor.

The default license for RealView Debugger enables you to debug applications that run on a single 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 Developer Suite v2.1 licensing* on page 1-8 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 Developer Suite v2.1 documentation* on page 1-9.

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, youmight 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 licenses. See *RealView Developer Suite v2.1 licensing* on page 1-8 for details.

For more details on multiprocessor debugging, see the chapter on working with multiple target connections in the *RealView Debugger Extensions User Guide*.

DSP debugging

RealView Debugger provides support for debugging the Oak and Teaklite DSP, and the Motorola M56621 DSP.

DSP debugging requires a separately purchased licenses. See *RealView Developer Suite v2.1 licensing* on page 1-8 for details.

For more details on DSP support, see the chapter on DSP support in the *RealView Debugger Extensions 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, or a Joint Test Action Group (JTAG) interface unit such as RealView ICE. Basic trace support is provided by RealView ARMulator ISS hardware simulator.

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

RTOS awareness

RTOS awareness is an extension that is built into RealView Debugger. You must obtain the plugin for the RTOS you are using before you can use this RealView Debugger extension. With an RTOS plugin, RealView Debugger references RTOS 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* for details on how to obtain the plugin for your RTOS.

RTOS awareness in RealView Debugger is vendor-independent. You can download and use RTOS plugins from more than one vendor. Therefore, you can develop applications for different OS platforms. If you want to use the RTOS 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 RTOS support, see the chapter on RTOS support in the *RealView Debugger Extensions User Guide*.

RealView Debugger downloads

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

Help → Goto RTOS Awareness Downloads

Displays the *RealView Debugger OS Awareness* web page. From here you can locate and download the RTOS plugin of your choice.

Help → Goto Board Awareness Downloads

Displays the *Board Awareness* web page.

Help → Goto Update and Utility Downloads

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

1.1.5 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® assembler, `armasm`
- ARM and Thumb C and C++ compiler, `armcc`
- 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 tools and utilities, and how to use them, see the RealView Compilation Tools documentation. The documentation is listed in *RealView Developer Suite v2.1 documentation* on page 1-9.

RVCT and RealView Debugger

You can use the RVCT tools from the command line interface (CLI). However, RealView Debugger provides a self-contained environment for building your applications, and then debugging them. If you want to create libraries or convert images to another format, you can configure RealView Debugger to automatically run the `armar` and `fromelf` utilities after a successful build. See the chapter on customizing projects in the *RealView Debugger Project Management User Guide* for details on customizing builds.

RealView Debugger also contains a built-in CLI. Apart from the debugging commands, you can also run native operating systems from this built-in CLI without having to open another window on your system. Therefore, you can build an image using simple build commands. To enter native operating system commands, use the RealView Debugger `HOST` command. You can also use this command in scripts. For example:

```
host armcc -g -c file1.c
host armlink file1.o -o file1.axf
```

Note

Be aware that RealView Debugger maintains a current working directory, and that you cannot change this directory using the `HOST` command.

For more details on RealView Debugger CLI commands, and the current working directory, see the *RealView Debugger Command Line Reference Guide*.

RVCT and armsd

armsd enables you to run native operating system commands. Therefore, you can run the RVCT build tools without exiting armsd first. To do this, use the ! command followed by the RVCT command, for example:

```
! armcc -g -c file1.c
! armlink file1.o -o file1.axf
```

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

1.1.6 RealView ARMulator Instruction Set Simulator

RealView ARMulator Instruction Set Simulator (RealView ARMulator ISS) 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 RealView ARMulator ISS interface connections that are available from the ARM debuggers on Windows, Sun Solaris, and Red Hat Linux. If you connect to RealView ARMulator ISS through RealView Connection Broker, you can connect to multiple instances of RealView ARMulator ISS.

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

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

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

1.2 RealView Developer Suite v2.1 licensing

All licensing for RealView Developer Suite v2.1 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.

The RealView Developer Suite v2.1 licenses that are separately available for RealView Debugger features are described in:

- *Multiprocessor debugging license*
- *Oak and Teaklite DSP debugging license*
- *Motorola M56621 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 1-4).

1.2.2 Oak and Teaklite DSP debugging license

The Oak and Teaklite *Digital Signal Processor* (DSP) support license enables you to debug applications running on Oak and Teaklite DSPs (see *DSP debugging* on page 1-4).

1.2.3 Motorola M56621 DSP debugging license

The Motorola M56621 DSP debugging license enables you to debug applications running on the Motorola M56621 DSP (see *DSP debugging* on page 1-4).

1.3 RealView Developer Suite v2.1 documentation

This section describes the documentation provided with RealView Developer Suite. It contains the following sections:

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

1.3.1 List of documents

The RealView Developer Suite documentation comprises:

- *ARM FLEXlm License Management Guide*
- *RealView Debugger Essentials Guide*
- *RealView Debugger User Guide*
- *RealView Debugger Project Management User Guide*
- *RealView Debugger Target Configuration Guide*
- *RealView Debugger Command Line Reference Guide*
- *RealView Debugger Extensions User 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 Developer Suite AXD and armsd Debuggers Guide*.

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

The full documentation suite is available online as PDF files. These are installed for a Typical installation, or if you choose to install them during a Custom installation. The documentation is installed in the documentation directory shown in Table 1-1 on page 1-2.

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.

On Windows, you can access the documentation from the Start menu:

Start → Programs → ARM

1.4 RealView Developer Suite Examples

The code for many of the examples in the RealView developer Suite documentation is located in the main examples directory (see *RealView Developer 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` This directory contains some examples of ARM assembly language programming. The examples are used in the *RealView Compilation Tools Assembler Guide*.

`cached_dhry` This directory contains examples of routines to initialize cache and TCMs, built around the Dhrystone example.

`cpp` This directory contains some simple C++ examples. In addition, the subdirectory `rw` contains the Rogue Wave manual and tutorial examples.

`databort` This directory contains design documentation (`dabort.txt`) and example code for a standard Data Abort handler.

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

`dhryansi` This directory contains an ANSI C version of Dhrystone.

`dhrystone` This directory contains the source for Dhrystone. The example is used in the RealView Debugger documentation.

`dsp` This directory contains a small source file that is required in order 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` This directory contains the example projects referenced in the chapter on embedded software developement in the *RealView Compilation Tools Developer Guide*. The following subdirectories are included:

- `buildn` Batch and make files to build the example projects.
- `dhry` Source files for the Dhrystone benchmarking program. This program provides the code base for the example projects in the individual `buildn` directories.
- `source` All other source files needed to build the example projects.

`include`	User defined header files.
`scatter`	Scatter files used to build the example projects.

`embedded` This directory is provided for backwards compatibility only, and contains source code examples that show how to write code for ROM. The examples are targeted at the ARM Integrator™ board. If you are writing new code with RVCT v2.0 and later, see the example in the `emb_sw_dev` directory.

`explasm` This directory contains additional ARM assembly language examples.

`fft_5te` Fast Fourier Transform code optimized for ARM architecture v5TE.

`inline` This directory contains examples that show how to use the inline assembler when compiling ARM C and C++ code. The examples are described in the chapter on mixing C, C++ and assembly language in the *RealView Compilation Tools Developer Guide*.

`interwork` This directory contains examples that show how to interwork between ARM code and Thumb code. The examples are described in the chapter on interworking ARM and Thumb in the *RealView Compilation Tools Developer Guide*.

`mmugen` This directory contains the source and documentation (`mmugen.pdf`) 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` This directory contains an example of how to write position-independent code.

`primes` This directory contains an example that demonstrates Trace support. The program calculates the nth prime number. However, as given, it fails with a Data Abort error at runtime.

`sorts` This directory contains example code that compares an insertion sort, shell sort, and the quick sort used in the ARM C libraries.

`swi` This directory contains an example SWI handler.

`unicode` This directory contains example code that enables you to evaluate the multibyte character support in RVCT v2.1.

`vfpsupport` This directory contains 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 98 VFP Support Code* (`AN98_vfp.pdf`).

1.5 ARM Developer Suite

RealView Developer Suite v2.1 also includes the full version of *ARM Developer Suite*™ (ADS) v1.2.1, for Windows only. ADS v1.2.1 is not installed as part of the RealView Developer Suite v2.1 installation. If you have to use ADS v1.2.1, you must install it separately.

Note

Although you can install ADS in addition to RealView Developer Suite v2.1, you must exercise caution if you use both the ADS ARMulator that is installed with ADS and RealView ARMulator ISS. See the *RealView Developer Suite v2.1 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).

1.5.1 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 vehicle support

The target vehicles supported by the ARM debuggers in the RVDS v2.1 and ADS v1.2.1 are shown in Table 1-3.

Table 1-3 Target vehicles supported by the ARM debuggers

Target vehicle	RVDS v2.1 support			ADS v1.2.1 support
	RealView Debugger	AXD v1.3	armsd	
Angel debug monitor	Yes	Yes	Yes	Yes (AXD only)
Agilent Debug Interface	Yes	Yes		Yes (AXD only)
Gateway		Yes		Yes (AXD only)
Multi-ICE	Yes	Yes		Yes (AXD only)
RealMonitor	Yes	Yes		Yes (AXD only)
RealView ARMulator ISS	Yes	RDI only	RDI only	RDI only
RealView ICE	Yes			
RealView Trace	Yes			

Chapter 2
Differences

This chapter summarizes the main differences between *RealView® Developer Suite* (RVDS) v2.1 and RVDS v2.0, and *ARM Developer Suite™* (ADS) v1.2.1. The changes are described in:

- *Changes between RVDS v2.1 and RVDS v2.0* on page 2-2
- *Changes between RVDS v2.1 and ADS v1.2.1* on page 2-4.

2.1 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*.

2.1.1 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 1-3)
- *ARM Symbolic Debugger* (armsd) is included (see *ARM Symbolic Debugger* on page 1-3)
- RealView Debugger has:
  - trace and profiling enhancements
  - RealView Debugger has enhanced RTOS support
  - RealView Debugger has new toolbar buttons and menu changes that mean you now have quick access to commonly used features.

  See *RealView Debugger* on page 1-3 for more information.

2.1.2 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.
- Many GNU C and C++ extensions are supported.
- Many new command-line options have been added to the build tools.

- The single-dash keyword options are deprecated.
- Many 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 *RealView Compilation Tools Linker and Utilities Guide* for more details.

See *RealView Compilation Tools* on page 1-6 for more information.

2.2 Changes between RVDS v2.1 and ADS v1.2.1

This section describes the changes between RealView Developer Suite v2.1 and ADS v1.2.1. It contains the following sections:

- *Debugger changes*
- *Build tool changes*
- *ARM simulator changes* on page 2-6.

2.2.1 Debugger changes

The main differences between the debugging tools in RVDS v2.1 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.
- AXD is enhanced to be able to debug C and C++ programs built with the new RealView Compilation Tools provided with RVDS v2.1.

See *RealView Debugger* on page 1-3 for more information.

2.2.2 Build tool changes

The main differences between the build tools in RVDS v2.1 and ADS v1.2.1 are:

- Compliance with the new ABI for the ARM Architecture (Base Standard). See the ABI for the ARM Architecture page at `http://www.arm.com/`.
- 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.
- Removal of unused C++ virtual functions.
- Multifile compilation, which performs optimizations across multiple compilation units.
- 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.
- Full support for ARM architecture v6 instructions has been added.
- Read/write data compression enables the optimization of ROM size.
- 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.
- The *ARM Profiler* (`armprof`) is not provided with RVCT.
- The ARM Applications Library is not provided with RVCT.
- 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.
- There are no temporary licenses.

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 *RealView Compilation Tools Linker and Utilities Guide* for more details.

See *RealView Compilation Tools* on page 1-6 for more information.

2.2.3 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 Developer Suite v2.1, you must exercise caution if you use both RealView ARMulator ISS and ADS ARMulator. See the *RealView Developer Suite v2.1 Release Notes* for more details.

See *RealView ARMulator Instruction Set Simulator* on page 1-7 for more information.

Glossary

The items in this glossary are listed in alphabetical order, with any symbols and numerics appearing at the end.

AAPCS *See* Procedure Call Standard for the ARM Architecture.

ABI for the ARM Architecture (Base Standard) (BSABI)
The ABI for the ARM Architecture is a collection of specifications, some open and some specific to ARM architecture, that regulate the inter-operation of binary code in a range of ARM architecture-based execution environments. The base standard specifies those aspects of code generation that must be standardized to support inter-operation and is aimed at authors and vendors of C and C++ compilers, linkers, and runtime libraries.

ADS *See* ARM Developer Suite.

American National Standards Institute (ANSI)
An organization that specifies standards for, among other things, computer software. This is superseded by the International Standards Organization.

Angel Angel is a software debug monitor that runs on the target and enables you to debug applications running on ARM architecture-based hardware. Angel is commonly used where a JTAG emulator is not available.

ANSI *See* American National Standards Institute.

APCS ARM Procedure Call Standard.

ARM Developer Suite (ADS)
A suite of software development applications, together with supporting documentation and examples, that enable you to write and debug applications for the ARM family of *RISC* processors. ADS is superseded by RealView Developer Suite (RVDS).

See also RealView Developer Suite.

ARM eXtended Debugger (AXD)
The *ARM eXtended Debugger* (AXD) is a single-processor debugger that runs on Windows platforms. AXD supports Multi-ICE and RealView ARMulator ISS debug targets.

See also ARM Symbolic Debugger, Multi-ICE, RealView ARMulator ISS, and RealView Debugger.

ARM Symbolic Debugger (armsd)
ARM Symbolic Debugger (armsd) is a command-line debugger that runs on all supported platforms. armsd supports only RealView ARMulator ISS debug targets.

See also ARM eXtended Debugger, RealView ARMulator ISS, and RealView Debugger.

armasm The ARM assembler.

armcc The ARM C compiler.

armlink The ARM C linker.

armsd *See* ARM Symbolic Debugger.

AXD *See* ARM eXtended Debugger.

Big-endian Memory organization where the least significant byte of a word is at the highest address and the most significant byte is at the lowest address in the word.

See also Little-endian.

Board RealView Developer Suite uses the term *board* to refer to a target processor, memory, peripherals, and debugger connection method.

Board file The *board file* is the top-level configuration file, normally called `rvdebug.brd`, that references one or more other files.

Coprocessor An additional processor used for certain operations. Usually used for floating-point math calculations, signal processing, or memory management.

Debug With Arbitrary Record Format (DWARF)
ARM code generation tools generate debug information in DWARF2 format.

Digital Signal Processor (DSP)
DSPs are special processors designed to execute repetitive, maths-intensive algorithms. Embedded applications might use both ARM processor cores and DSPs.

DSP *See* Digital Signal Processor.

DWARF *See* Debug With Arbitrary Record Format.

ELF *See* Executable and Linking Format.

Embedded Trace Macrocell (ETM)
A block of logic, embedded in the hardware, that is connected to the address, data, and status signals of the processor. It broadcasts branch addresses, and data and status information in a compressed protocol through the trace port. It contains the resources used to trigger and filter the trace output.

EmbeddedICE logic The EmbeddedICE logic is an on-chip logic block that provides TAP-based debug support for ARM processor cores. It is accessed through the TAP controller on the ARM core using the JTAG interface.

See also IEEE1149.1.

ETM *See* Embedded Trace Macrocell.

ETV *See* Extended Target Visibility.

Executable and Linking Format (ELF)
Executable and Linking Format (ELF) is a format used by the ARM code generation tools to produce objects and executable images.

Execution vehicle Part of the debug target interface, execution vehicles process requests from the client tools to the target.

Extended Target Visibility (ETV)
Extended Target Visibility enables RealView Developer Suite to access features of the underlying target, such as chip-level details provided by the hardware manufacturer or SoC designer.

Image An execution file that has been loaded onto a processor for execution.

Integrator A range of ARM hardware development platforms. *Core modules* are available that contain the processor and local memory.

Joint Test Action Group (JTAG)
An IEEE group focussed on silicon chip testing methods. Many debug and programming tools use a *Joint Test Action Group* (JTAG) interface port to communicate with processors. For further information refer to IEEE Standard, Test Access Port and Boundary-Scan Architecture specification 1149.1 (JTAG).

JTAG *See* Joint Test Action Group.

JTAG interface unit A protocol converter that converts low-level commands from RealView Developer Suite into JTAG signals to the EmbeddedICE logic and the ETM.

Little-endian Memory organization where the least significant byte of a word is at the lowest address and the most significant byte is at the highest address of the word.

See also Big-endian.

Multi-ICE A JTAG-based tool for debugging embedded systems.

Procedure Call Standard for the ARM Architecture (AAPCS)
The Procedure Call Standard for the ARM Architecture specifies a family of *Procedure Call Standard* (PCS) variants, to define how separately compiled and assembled routines can work together. The standard provides equal support for both ARM state and Thumb state to enable interworking. It favors small code-size, and provides functionality appropriate to embedded applications and high performance.

Profiling Accumulation of statistics during execution of a program being debugged, to measure performance or to determine critical areas of code.

RDI *See* Remote Debug Interface.

RealView ARMulator ISS (RVISS)
The most recent version of the ARM simulator, RealView ARMulator ISS is supplied with RealView Developer Suite. It communicates with a debug target using RV-msg, through the RealView Connection Broker interface, and RDI.

See also RDI and RealView Connection Broker.

RealView Compilation Tools (RVCT)
RealView Compilation Tools is a suite of tools, together with supporting documentation and examples, that enables you to write and build applications for the ARM family of *RISC* processors.

RealView Connection Broker
RealView Connection Broker is an execution vehicle that enables you to connect to simulator targets on your local system, or on a remote system. It also enables you to make multiple connections to the simulator.

See also RealView ARMulator ISS.

RealView Debugger
RealView Debugger is a multiprocessor debugger that runs on all supported platforms. You can connect to debug targets using Multi-ICE, RVISS, and RealView ICE.

See also ARM Symbolic Debugger, Multi-ICE, RealView ARMulator ISS, and RealView ICE.

RealView Debugger Trace
Part of RealView Debugger that extends the debugging capability with the addition of real-time program and data tracing.

Remote Debug Interface (RDI)
The *Remote Debug Interface* (RDI) is an ARM standard procedural interface between a debugger and the debug agent. RDI gives the debugger a uniform way to communicate with:

- a simulator running on the host (for example, RVISS)
- a debug monitor running on ARM architecture-based hardware accessed through a communication link (for example, Angel)
- a debug agent controlling an ARM processor through hardware debug support (for example, Multi-ICE).

RealView Developer Suite
The latest suite of software development applications from ARM Limited, together with supporting documentation and examples, that enable you to write and debug applications for the ARM family of *RISC* processors.

See also ARM Developer Suite.

RealView ICE (RVI) A JTAG-based debug solution to debug software running on ARM processors.

RISC Reduced Instruction Set Computer.

RTOS Real Time Operating System.

RVCT *See* RealView Compilation Tools.

RVISS *See* RealView ARMulator ISS.

Simulator A simulator executes non-native instructions in software (simulating a core).

See also RealView ARMulator ISS.

Target The target hardware, including processor, memory, and peripherals, real or simulated, on which the target application is running.

Target Vehicle Server (TVS)
Essentially the RealView Debugger itself, this contains the basic debugging functionality. TVS contains the run control, base multitasking support, much of the command handling, target knowledge, such as memory mapping, lists, rule processing, board-files and `.bcd` files, and data structures to track the target environment.

Tracing The real-time recording of processor activity (including instructions and data accesses) that occurs during program execution. Trace information can be stored either in a trace buffer of a processor, or in an external trace hardware unit. Captured trace information is returned to the Analysis window in RealView Debugger where it can be analyzed to help identify a defect in program code.

TVS *See* Target Vehicle Server.

Index

The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.

P

Q

R

S

T

