ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *ARM MultiTrace / Multi-ICE* at our website: *Click HERE*

Multi-ICE®

Version 2.2

User Guide

ARM®

Copyright © 1998-2002 ARM Limited. All rights reserved.
ARM DUI 0048F

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Multi-ICE
User Guide

Copyright © 1998-2002 ARM® Limited. All rights reserved.

Release Information

The following changes have been made to this document.

Change history

Date	Issue	Change
June 1998	A	First release
November 1998	B	Internal release
December 1998	C	Updated for Multi-ICE Release 1.3
January 2001	D	Updated for Multi-ICE Version 2.0
September 2001	E	Updated for Multi-ICE Version 2.1
February 2002	F	Updated for Multi-ICE Version 2.2

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by ARM Limited. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Web Address

`http://www.arm.com`

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Conformance Notices

This section contains *ElectroMagnetic Conformity* (EMC) notices and other important notices.

Federal Communications Commission Notice

This device is test equipment and consequently is exempt from part 15 of the FCC Rules under section 15.103 (c).

CE Declaration of Conformity

This equipment has been tested according to ISE/IEC Guide 22 and EN 45014. It conforms to the following product EMC specifications:

The product herewith complies with the requirements of EMC Directive 89/336/EEC as amended.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents
Multi-ICE User Guide

Preface
About this document xiv
Feedback xx

Chapter 1 Introduction
1.1 About Multi-ICE 1-2
1.2 Availability and compatibility 1-3
1.3 Basic principles 1-4
1.4 Introduction to the Multi-ICE components 1-7
1.5 New features and changes from previous versions 1-11

Chapter 2 Getting Started
2.1 System requirements 2-2
2.2 Connecting the Multi-ICE hardware 2-6
2.3 Connecting to nonstandard hardware 2-11
2.4 Starting the software 2-14

Chapter 3 Using the Multi-ICE Server
3.1 About the Multi-ICE server menus 3-2
3.2 Multi-ICE server device configuration files 3-9
3.3 Server configuration 3-14
3.4 Using the Multi-ICE server with multiple processors 3-25

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 4 Debugging with Multi-ICE
4.1 Compatibility with ARM debuggers 4-2
4.2 Connecting Multi-ICE to ADW, ADU, or AXD 4-3
4.3 Configuring the Multi-ICE DLL 4-8
4.4 Configuring and debugging multiple processors 4-27
4.5 Debugger internal variables 4-36
4.6 Post-mortem debugging 4-45
4.7 Access to CP15 4-49
4.8 Semihosting 4-50
4.9 Watchpoints and breakpoints 4-55
4.10 Cached data 4-60
4.11 Debugging applications in ROM 4-62
4.12 Accessing the EmbeddedICE logic directly 4-65

Chapter 5 Troubleshooting
5.1 Troubleshooting 5-2
5.2 Error messages 5-12

Chapter 6 System Design Guidelines
6.1 About the system design guidelines 6-2
6.2 System design 6-3
6.3 ASIC guidelines 6-9
6.4 PCB guidelines 6-12
6.5 JTAG signal integrity and maximum cable lengths 6-15
6.6 Compatibility with EmbeddedICE interface target connectors 6-17

Appendix A Server Configuration File Syntax
A.1 IR length configuration file A-2
A.2 Device configuration file A-3

Appendix B Breakpoint Selection Algorithm
B.1 Multi-ICE internal breakpoints B-2
B.2 How the debugger steps and runs code B-4
B.3 Breakpoint and watchpoint allocation algorithm B-5

Appendix C Command-line Syntax
C.1 Multi-ICE server C-2

Appendix D Processor-specific Information
D.1 The ARM1020T (Rev 0) processor D-2
D.2 Intel XScale microarchitecture processors D-3

Appendix E CP15 Register Mapping
E.1 About register mapping E-2
E.2 ARM710T processor registers E-3

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.3 ARM720T processor registers E-4
E.4 ARM740T processor registers E-5
E.5 ARM920T and ARM922T processor registers E-6
E.6 ARM925T processor registers E-10
E.7 ARM926EJ-S processor registers E-14
E.8 ARM940T processor registers E-18
E.9 ARM946E-S processor registers E-21
E.10 ARM1020T and ARM10200T processor registers E-24
E.11 XScale microarchitecture processor registers E-28

Appendix F JTAG Interface Connections
F.1 Multi-ICE JTAG interface connections F-2
F.2 Multi-ICE JTAG port timing characteristics F-5
F.3 TCK frequencies F-7
F.4 TCK values F-11

Appendix G User I/O Connections
G.1 Multi-ICE user I/O pin connections G-2

Glossary

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Tables
Multi-ICE User Guide

	Change history	ii
Table 2-1	Supported operating systems for Multi-ICE	2-2
Table 3-1	TCK frequency for autoconfigure	3-3
Table 3-2	Scale and multiplier values	3-23
Table 3-3	Scale values for clocking speeds	3-23
Table 4-1	ARM7 family debugger variable support	4-37
Table 4-2	ARM9 family debugger variable support	4-38
Table 4-3	ARM10 family and XScale microarchitecture debugger variable support	4-39
Table 4-4	Cache selection type values	4-40
Table 4-5	Breakpoints	4-57
Table E-1	ARM710T processor registers	E-3
Table E-2	ARM720T processor registers	E-4
Table E-3	ARM740T processor registers	E-5
Table E-4	ARM920T and ARM922T processor registers	E-6
Table E-5	ARM920T and ARM922T cp15 register 7 accesses	E-8
Table E-6	ARM920T and ARM922T cp15 register 8 accesses	E-9
Table E-7	ARM925T processor registers	E-10
Table E-8	ARM925T cp15 register 7 accesses	E-11
Table E-9	ARM925T cp15 register 8 accesses	E-12
Table E-10	ARM925T cp15 register 7 accesses	E-13
Table E-11	ARM926EJ-S processor registers	E-14
Table E-12	ARM926EJ-S cp15 register 7 accesses	E-16
Table E-13	ARM926EJ-S cp15 register 8 accesses	E-17

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-14	ARM940T processor registers	E-18
Table E-15	ARM940T cp15 register 7 accesses	E-19
Table E-16	ARM946E-S processor registers	E-21
Table E-17	ARM946E-S cp15 register 7 accesses	E-23
Table E-18	ARM1020T and ARM10200T processor registers	E-24
Table E-19	ARM1020T and ARM10200T cp15 register 7 accesses	E-25
Table E-20	ARM1020T and ARM10200T cp15 register 8 accesses	E-26
Table F-1	JTAG pinouts	F-3
Table F-2	Multi-ICE IEEE 1149.1 timing requirements	F-5
Table F-3	TCK frequencies	F-7
Table F-4	TCK values	F-11
Table G-1	User I/O connections	G-2

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Figures
Multi-ICE User Guide

	Key to timing diagram conventions	xvii
Figure 1-1	The Multi-ICE interface unit	1-7
Figure 1-2	Connecting multiple debuggers and multiple targets	1-9
Figure 2-1	The Multi-ICE product kit	2-7
Figure 2-2	Multi-ICE interface unit cable connection	2-8
Figure 2-3	Location of jumper J8	2-9
Figure 2-4	Multi-ICE current consumption with voltage	2-13
Figure 2-5	Start menu items for Multi-ICE	2-14
Figure 2-6	Unconfigured Multi-ICE server window	2-16
Figure 2-7	Multi-ICE server window configured for an ARM7TDMI	2-17
Figure 3-1	Multi-ICE server menu items	3-2
Figure 3-2	The File menu	3-3
Figure 3-3	The View menu	3-5
Figure 3-4	The Run Control menu	3-6
Figure 3-5	The Connection menu	3-7
Figure 3-6	The Settings menu	3-7
Figure 3-7	The Help menu	3-8
Figure 3-8	Autoconfiguring an ARM940T	3-10
Figure 3-9	TAP driver status dialog	3-15
Figure 3-10	TAP Controller Device ID register format	3-16
Figure 3-11	The Start-up Options dialog	3-16
Figure 3-12	The Port Settings dialog	3-18
Figure 3-13	The User Output Bits dialog	3-20

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 3-14 Status of the user input bits 3-21
Figure 3-15 The JTAG Settings dialog 3-21
Figure 3-16 Setting up interaction between devices 3-28
Figure 3-17 Cascade operation 3-29
Figure 3-18 Setting up the poll frequency 3-30
Figure 4-1 The AXD Options menu 4-4
Figure 4-2 The AXD Choose Target dialog 4-4
Figure 4-3 Selecting the Multi-ICE DLL using AXD 4-5
Figure 4-4 The ADW and ADU Options menu 4-5
Figure 4-5 ADW configuration dialog with Multi-ICE active 4-6
Figure 4-6 Selecting the Multi-ICE DLL using ADW 4-6
Figure 4-7 Multi-ICE Configuration dialog 4-8
Figure 4-8 Multi-ICE Welcome dialog 4-9
Figure 4-9 Driver Details dialog 4-10
Figure 4-10 Server Browse dialog 4-12
Figure 4-11 Multi-ICE Processor Settings tab showing cache setting 4-15
Figure 4-12 Multi-ICE Processor Settings tab showing XScale settings 4-16
Figure 4-13 Multi-ICE Advanced settings tab 4-18
Figure 4-14 Board tab 4-21
Figure 4-15 Trace configuration tab 4-22
Figure 4-16 About tab 4-23
Figure 4-17 Channel viewer controls 4-24
Figure 4-18 Saving a named target configuration 4-28
Figure 4-19 Configuring AXD to run a configuration script 4-31
Figure 4-20 Three AXDs and the Multi-ICE server configured for a multiple processor target .. 4-32
Figure 4-21 Relating top_of_memory to single section program layout 4-43
Figure 4-22 Register view showing EmbeddedICE logic registers 4-66
Figure 4-23 The View Registers menu 4-67
Figure 4-24 The Display Co-processor Regs dialog 4-67
Figure 4-25 EmbeddedICE logic registers in the Raw Co-processor 0 view 4-68
Figure 6-1 Basic JTAG port synchronizer 6-4
Figure 6-2 Timing diagram for the Basic JTAG synchronizer in Figure 6-1 on page 6-4 6-5
Figure 6-3 JTAG port synchronizer for single rising-edge D-type ASIC design rules 6-5
Figure 6-4 Timing diagram for the D-type JTAG synchronizer in Figure 6-3 on page 6-5 6-6
Figure 6-5 Example reset circuit logic 6-8
Figure 6-6 Example reset circuit using power supply monitor ICs 6-8
Figure 6-7 TAP Controllers serially chained in an ASIC 6-10
Figure 6-8 Typical PCB connections 6-12
Figure 6-9 Target interface voltage levels 6-13
Figure F-1 JTAG pin connections, top view F-2
Figure F-2 Multi-ICE JTAG port timing diagram F-5
Figure G-1 User I/O pin connections G-2
Figure G-2 Converting user-input signals to TTL levels G-4

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Preface

This preface introduces the *Multi-ICE Version 2.2 User Guide*. It explains the structure of the user guide and lists other sources of information that relate to Multi-ICE and ARM debuggers.

This preface contains the following sections:

- *About this document* on page xiv
- *Feedback* on page xx.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

About this document

This document describes the ARM *MULTI-processor embeddedICE interface unit* (Multi-ICE) Version 2.2.

Intended audience

This document is written for users of Multi-ICE on Windows or Unix platforms, using either the ARM *Software Development Toolkit* (SDT) or *ARM Developer Suite* (ADS) development environments. It is assumed that you are a software engineer with some experience of the ARM architecture, or a hardware engineer designing a product that is compatible with Multi-ICE.

Parts of this document assume you have some knowledge of JTAG technology. If you require more information on JTAG, refer to *IEEE Standard 1149.1*, available from the *Institute of Electrical and Electronic Engineers* (IEEE). Refer to the IEEE website for more information at:

`http://www.ieee.org/`

Organization

This document is organized into the following chapters and appendices:

Chapter 1 *Introduction*

Read this chapter for a description of:

- what is provided in the Multi-ICE product
- the purpose of the EmbeddedICE® logic within the CPU
- what has changed between Multi-ICE Version 2.2 and Version 2.1, between Multi-ICE Version 2.1 and Version 2.0, between Version 2.0 and Release 1.4, and between Release 1.4 and Release 1.3.

Chapter 2 *Getting Started*

Read this chapter for information on how to start working with Multi-ICE. The chapter includes the hardware and software system requirements, how to connect up the hardware, and how to start the Multi-ICE server.

Chapter 3 *Using the Multi-ICE Server*

This chapter describes how you use the Multi-ICE server, including a more detailed description of configuring the server. There are also sections describing the execution control and user I/O features.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 4 *Debugging with Multi-ICE*

This chapter describes how to:

- connect Multi-ICE to an ARM debugger
- change the behavior of Multi-ICE using internal variables
- implement watchpoints and breakpoints and what this means to you
- access the EmbeddedICE logic directly.

You must read this chapter in conjunction with the debugger user documentation, for example the *ADS Debuggers Guide*.

Chapter 5 *Troubleshooting*

Read this chapter for a troubleshooting guide and a list of error messages.

Chapter 6 *System Design Guidelines*

Read this chapter for information about designing ARM-based ASICs and PCBs that can be debugged using Multi-ICE.

It includes:

- suggested clocking and reset circuit diagrams
- how to chain TAP controllers
- suggested physical connector types and pinouts
- a description of logic voltage level adaption
- how power consumption varies with supply voltage.

Appendix A *Server Configuration File Syntax*

This appendix describes the server configuration file. This file describes a target device group to Multi-ICE.

Appendix B *Breakpoint Selection Algorithm*

This appendix describes how Multi-ICE allocates your breakpoints and internally generated breakpoints to the hardware. You must read it if you require specific types of breakpoint to be allocated to target memory regions.

Appendix C *Command-line Syntax*

This appendix describes the command-line syntax of the Multi-ICE server.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix D ***Processor-specific Information***

This appendix describes the differences in the way that Multi-ICE behaves on the ARM10 and XScale microarchitecture processors.

Appendix E ***CP15 Register Mapping***

This appendix contains details relating to register mapping information for the ARM7-based, ARM9-based, ARM10-based, and XScale processors containing a system control coprocessor (CP15).

Appendix F ***JTAG Interface Connections***

This appendix describes and illustrates the JTAG pin connections.

Appendix G ***User I/O Connections***

This appendix describes and illustrates the additional input and output connections provided in Multi-ICE.

Typographical conventions

The following typographical conventions are used in this document:

bold — Highlights ARM processor signal names within text, and interface elements such as menu names. Can also be used for emphasis in descriptive lists where appropriate.

italic — Highlights special terminology, cross-references and citations.

`monospace` — Denotes text that can be entered at the keyboard, such as commands, file names 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.

`typewriter italic`

Denotes arguments to commands or functions where the argument is to be replaced by a specific value.

`typewriter bold`

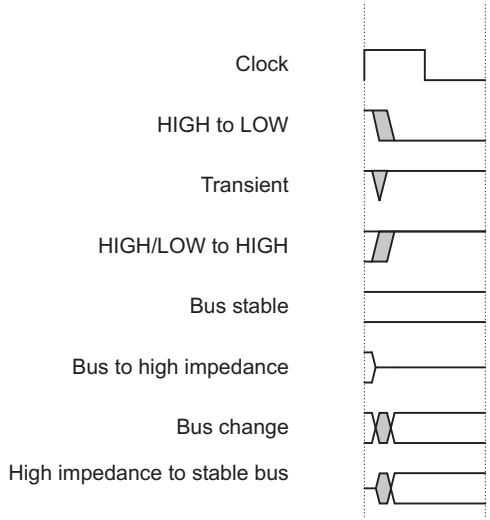
Denotes language keywords when used outside example code.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Timing diagram conventions

This manual contains timing diagrams. The following diagram shows the components used in these diagrams. Any variations are clearly labeled when they occur. Therefore, no additional meaning must be attached unless specifically stated.



Key to timing diagram conventions

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

Further reading

This section lists publications by ARM Limited, and by third parties, that are related to this product.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com/arm/documentation` for current errata sheets and addenda.

See also the ARM Frequently Asked Questions list at: `http://www.arm.com/arm/tech_faqs`.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ARM publications

This document contains information that is specific to Multi-ICE. The following documents also relate specifically to Multi-ICE:

- *Multi-ICE TAPOp API Reference Guide* (ARM DUI 0154)
- *ARM Multi-ICE Installation Guide* (ARM DSI 0005)
- Multi-ICE file `Readme.txt`, supplied on the Multi-ICE distribution CD and installed with the product
- Multi-ICE file `proclist.txt`, a list of the processors supported by Multi-ICE and installed with the product.

If you are using Multi-ICE with the *ARM Developer Suite* (ADS) v1.2, refer to the following books in the ADS document suite for information on other components of ADS:

- *Installation and License Management Guide* (ARM DUI 0139)
- *Getting Started* (ARM DUI 0064)
- *CodeWarrior IDE Guide* (ARM DUI 0065)
- *AXD and armsd Debuggers Guide* (ARM DUI 0066)
- *Compilers and Libraries Guide* (ARM DUI 0067)
- *Linker and Utilities Guide* (ARM DUI 0151)
- *Assembler Guide* (ARM DUI 0068)
- *Developer Guide* (ARM DUI 0056)
- *Debug Target Guide* (ARM DUI 0058)
- *Trace Debug Tools User Guide* (ARM DUI 0118)
- *ARM Application Library Programmers Guide* (ARM DUI 0081).

The following additional documentation that might be useful is provided with the ARM Developer Suite:

- *ARM Architecture Reference Manual* (ARM DDI 0100). This is supplied in Dynatext format as part of the online books, and as a PDF file.

In addition, refer to the following documentation for specific information relating to ARM products:

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

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Other publications

The following publications might also be useful to you, and are available from the indicated sources:

- *The Intel® XScale™ Core Developer's Manual*, Datasheet, advance information. Ref 27341401-002. Intel Corp. 2000.
- *Hot-Debug for Intel XScale Core Debug*, White paper. Ref 273539-002. Intel Corp. 2001.
- *IEEE Standard Test Access Port and Boundary Scan Architecture* (IEEE Std. 1149.1) describes the JTAG ports with which Multi-ICE communicates.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Feedback

ARM Limited welcomes feedback both on Multi-ICE and on the documentation.

Feedback on Multi-ICE

If you have any problems with Multi-ICE, please contact your supplier. To help us provide a rapid and useful response, please give:

- the Multi-ICE version you are using
- details of the platforms you are using, including both the host and target hardware types and operating system
- where appropriate, 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
- if possible, sample output illustrating the problem

Feedback on this document

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

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

General suggestions for additions and improvements are also welcome.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 1
Introduction

This chapter introduces Multi-ICE, and describes its software components and documentation. It contains the following sections:

- *About Multi-ICE* on page 1-2
- *Availability and compatibility* on page 1-3
- *Basic principles* on page 1-4
- *Introduction to the Multi-ICE components* on page 1-7
- *New features and changes from previous versions* on page 1-11.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.1 About Multi-ICE

Multi-ICE is the EmbeddedICE logic debug solution from ARM. It enables you to debug software running on ARM processor cores that include the EmbeddedICE logic.

Multi-ICE provides the software and hardware interface between a *Joint Test Action Group* (JTAG) *IEEE Standard 1149.1* port on the hardware using a small interface unit and a Windows or UNIX debugger using the ARM *Remote Debug Interface* (RDI) running on a workstation.

You can use Multi-ICE with systems that contain one or more ARM CPUs or DSP processors. It also supports the *Embedded Trace Macrocell* (ETM). The external hardware and software to decode this information is available separately from ARM Limited.

The Multi-ICE product comprises:

- An interface unit that connects the parallel port of a workstation to the JTAG interface of an ASIC that includes debug and EmbeddedICE capability.
- A cable to connect the interface unit to a parallel port. (This is underneath the foam packaging.)
- A 20-way ribbon cable. This connects the Multi-ICE interface unit to the target.
- Software on CD-ROM that enables an ARM debugger to communicate with the interface unit. The software includes the following components:
  - the Multi-ICE server
  - a *Dynamic Link Library* (DLL) to use with the debugger
  - documentation in PDF and Dynatext formats
  - example programs demonstrating the TAPOp *Application Program Interface* (API).
- Documentation, including:
  - a printed copy of this User Guide
  - the installation CD insert
  - a warranty notice and end user license.

A 20-way to 14-way cable adaptor for use with targets that use a 14-way target connection is available on request from ARM, part number HPI-0027.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2 Availability and compatibility

Multi-ICE is available from ARM Limited and its resellers as a package that includes:

- a JTAG hardware interface
- a software interface component that connects, using RDI, to an external debugger, supplied separately.

Upgrades from earlier versions of Multi-ICE are available. Please contact your reseller or check on the ARM Limited website for details.

The ADS Version 1.2 product includes the following fully supported debuggers:

- *ARM eXtended Debugger* (AXD, both Windows and UNIX versions)
- *ARM Debugger for Windows* (ADW)
- *ARM Debugger for UNIX* (ADU).

Multi-ICE is also compatible with third-party debuggers that conform to the ARM standard RDI 1.5.1 interface.

Contact ARM Limited directly regarding OEM licenses.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.3 Basic principles

The EmbeddedICE logic and the ARM processor debug extensions enable Multi-ICE to debug software running on an ARM processor. The following topics are covered:

- *Debug extensions to the ARM core*
- *The EmbeddedICE logic*
- *The ICE extension unit* on page 1-5
- *How Multi-ICE differs from a debug monitor* on page 1-5.

Note

To determine whether a specific ARM processor has support for JTAG debugging, refer to the datasheet or technical reference manual.

1.3.1 Debug extensions to the ARM core

The extensions consist of a number of scan chains around the processor core and some additional signals that are used to control the behavior of the core for debug purposes. The most significant of these additional signals are:

BREAKPT This core signal enables external hardware to halt processor execution for debug purposes. When HIGH, the current memory access is tagged as breakpointed and the core stops when this instruction is executed.

DBGRQ This core signal is a level-sensitive input that causes the CPU core to enter debug state when the current instruction has completed.

DBGACK This core signal is an output from the CPU core that goes HIGH when the core is in debug state allowing external devices to determine the current state of the core.

Multi-ICE uses these, and other signals, by using the debug interface of the processor core, for example by writing to the control register of the EmbeddedICE logic. For more details, refer to the debug interface section of the ARM datasheet or technical reference manual for your core.

1.3.2 The EmbeddedICE logic

The EmbeddedICE logic is the integrated on-chip logic that provides JTAG debug support for ARM cores. EmbeddedICE/RT is a superset of EmbeddedICE that includes extensions supporting real-time debug, including setting breakpoints on a running target.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The EmbeddedICE logic is accessed through the TAP controller on the ARM core using the JTAG interface. See Chapter 6 *System Design Guidelines* for details of designing this into your own target.

The standard EmbeddedICE logic consists of:

- two watchpoint units
- a control register
- a status register
- a set of registers implementing the Debug Communications Channel link.

For more details on the *Debug Communications Channel* (DCC), see the *ADS Developer Guide*.

You can program one or both of the watchpoint units to halt the execution of instructions by the ARM CPU core. Execution is halted when a match occurs between the values in the watchpoint registers and the values currently appearing on the address bus, data bus, and selected control signals.

You can mask any bit to prevent it from affecting the comparison. Either watchpoint unit can be configured to be a watchpoint (monitoring data accesses) or a breakpoint (monitoring instruction fetches).

For more information, refer to the relevant section of the appropriate ARM datasheet or a technical reference manual.

1.3.3 The ICE extension unit

The *ICE Extension Unit* (IEU) is a logic block that can be added to the EmbeddedICE logic when a processor is fabricated. The IEU extends the number of hardware breakpoint units available to the debugger using extra comparators and an extra JTAG scan chain. It is available in different sizes, providing up to 30 additional units. Multi-ICE supports the logic automatically on processors that include it.

1.3.4 How Multi-ICE differs from a debug monitor

A debug monitor, such as the Angel™ debug monitor provided with the *ARM Firmware Suite* (AFS), is an application that runs on your target hardware in conjunction with your application, and requires target resources (for example, memory, access to exception vectors, and time) to be available.

The EmbeddedICE debug architecture requires almost no resources. Rather than being an application on the board, it works by using:

- additional debug hardware within the core, that is, parts that enable the host to communicate with the target

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- an external interface unit that buffers and translates the core signals into something usable by a host computer.

The EmbeddedICE debug architecture allows debugging to be as non-intrusive as possible:

- the target being debugged requires very little special hardware to support debugging
- in most cases you do not have to set aside memory for debugging in the system being debugged and you do not have to incorporate special software into the application
- execution of the system being debugged is only halted when a breakpoint or watchpoint unit is triggered, or you request that execution is halted.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.4 Introduction to the Multi-ICE components

This section introduces the components of the Multi-ICE product, and describes how they fit together:

- *The Multi-ICE interface unit*
- *The Multi-ICE parallel port driver*
- *The Multi-ICE server* on page 1-8
- *The portmap application* on page 1-9.

If you are using the Multi-ICE DLL on a UNIX workstation, you must connect that workstation to the Windows workstation running the server using a TCP/IP network.

For more information about the Multi-ICE server see Chapter 2 *Getting Started* and Chapter 3 *Using the Multi-ICE Server*. For more information about the Multi-ICE Debugger interface see Chapter 4 *Debugging with Multi-ICE*.

1.4.1 The Multi-ICE interface unit

The Multi-ICE interface unit provides the hardware to allow a Windows workstation to control multiple JTAG capable devices. The unit is shown in Figure 1-1. The unit has a parallel port at one end, and a 20-pin JTAG connector and external power input at the other. A cable is supplied to connect the interface unit to the workstation parallel port.



Figure 1-1 The Multi-ICE interface unit

The interface unit gives the Windows workstation direct control of basic JTAG operations. This means new debugging features and support for new processors can be added with software updates.

1.4.2 The Multi-ICE parallel port driver

Although the Multi-ICE interface unit connects to your Windows workstation through a standard parallel port, Multi-ICE does not use the standard parallel port driver. Instead, an optimized driver is used that allows high-speed communication with the interface unit. This driver is installed automatically when you install Multi-ICE, and is configured using options in the Multi-ICE server. It is only required on the Windows workstation that runs the Multi-ICE server.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To access this driver:

- On Windows NT 4.0:
  1. Choose **Start** → **Settings** → **Control Panel**.
  2. Double-click on the **Devices** icon.
  3. In the list of devices, select the **Multi-ICE** driver.
- On Windows 2000:
  1. Choose **Start** → **Settings** → **Control Panel**.
  2. Double-click on the **System** icon.
  3. Select the **Hardware** tab.
  4. Click on the **Device Manager** button.
  5. Choose **View** → **Show hidden devices**.
  6. Expand the list of **Non-Plug and Play Drivers**.
  7. Select the **Multi-ICE** driver.
- On the other supported variants of Windows, you cannot access the driver.

1.4.3 The Multi-ICE server

The Multi-ICE server is an application that runs on the Windows workstation connected to the interface unit. The Multi-ICE server can address each JTAG device individually, without affecting other devices on the board. It uses this ability to create virtual connections for each of the JTAG devices on the board. Debugging software can attach to one of these virtual connections, and perform debugging operations with no knowledge of the other devices on the board, as shown in Figure 1-2 on page 1-9.

The Multi-ICE server enables multiple concurrent connections, so if you have a target with multiple processors, you can run several debuggers and connect each one to a different processor on the board. This allows you to easily debug multiprocessor systems. The server can also perform a synchronized start or stop of processors, for debugging multiprocessor systems where the processors interact with each other.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

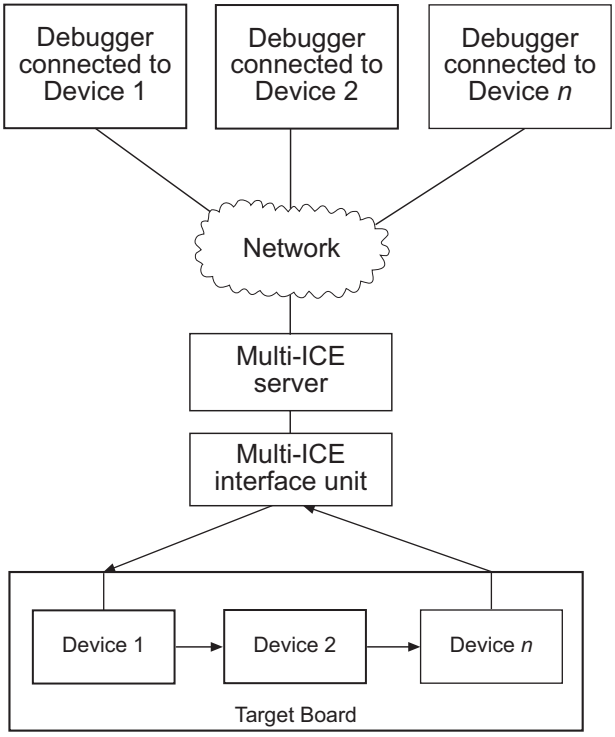


Figure 1-2 Connecting multiple debuggers and multiple targets

The Multi-ICE server also supports connections across a network, so the debugging software can be run on a different computer to the server, or on several different computers if that is appropriate.

1.4.4 The portmap application

To support network connections, an additional application must be running on the Windows workstation that runs the Multi-ICE server. This application is called the portmapper, and allows software on other computers on the network to locate the Multi-ICE server.

The portmapper is normally started automatically by the Multi-ICE server. It runs in a minimized console window, and requires no intervention by you. You can change the Multi-ICE server settings to prevent the portmapper being started if you do not require network connections, or if you already have another portmapper running on your Windows workstation.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.4.5 The Multi-ICE DLL

The Multi-ICE DLL is a software module that translates debugger commands, for example, start, stop, and download, into JTAG control sequences for a particular processor. It fits between a debugger, for example AXD, providing the user interface and the Multi-ICE server controlling the JTAG devices.

The Multi-ICE DLL is supplied in the form of a Windows or UNIX dynamically linked library. The Multi-ICE DLL provides support for debugging on a wide range of ARM cores (see `proclist.txt` for a list of supported cores). It can also be used with any graphical debugger that supports the RDI 1.5.1 interface. This includes all current ARM graphical debuggers (AXD, ADW, and ADU)

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.5 New features and changes from previous versions

This section describes the new features and other changes to the product:

- *New features in Multi-ICE Version 2.2*
- *Changes in Multi-ICE Version 2.2* on page 1-12
- *New features in Multi-ICE Version 2.1* on page 1-12
- *Changes in Multi-ICE Version 2.1* on page 1-13
- *New features in Multi-ICE Version 2.0* on page 1-13
- *Changes in Multi-ICE Version 2.0* on page 1-14
- *New features in Release 1.4* on page 1-14.

1.5.1 New features in Multi-ICE Version 2.2

The new features in Multi-ICE Version 2.2 are:

New processor support

ARM7 cores ARM7TDMI-S™ (Rev 4), ARM7EJ-S™ (Rev 1), and Samsung S3C4510B.

ARM9 cores ARM926EJ-S™ (Rev 0), and ARM966E-S™ (Rev 2).

Intel XScale microarchitecture

Intel PXA210, Intel PXA250, and Intel 80321.

New operating system support

Redhat Linux 6.2 and 7.1, Solaris 8.0, and HP-UX 11.

eXDI support

A driver that maps the Microsoft eXDI interface to the ARM RDI protocol is now bundled with Multi-ICE. Using this driver, you can connect to Multi-ICE from Microsoft Platform Builder.

For more information, see the *Multi-ICE Installation Guide*, and the online documentation that is supplied with the driver.

Support for peripherals and microcontrollers

An extra tab has been added to the configuration dialog for the Multi-ICE DLL. You can use this tab to select a specific target board. The debugger then shows the registers for the peripherals and microcontrollers on the selected board. For more information, see *Board configuration tab* on page 4-21.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Updated processor description files

Updated processor description files are supplied for use with ADS versions 1.1 and 1.2.

1.5.2 Changes in Multi-ICE Version 2.2

Changes in Multi-ICE Version 2.2 are:

XScale processor detection

The Intel 80200 processor is now detected as an `XScale-80200` processor, rather than as an `XScale` processor. Other XScale processors are now detected.

XScale performance counter behavior

The behavior of XScale performance counters in debug state has been slightly changed, to provide better support of Multi-ICE trace. See *Performance counters* on page D-6 for more information.

1.5.3 New features in Multi-ICE Version 2.1

The new features in Multi-ICE Version 2.1 are:

New processor support

ARM9 cores ARM922T™ (Rev 0), ARM925T™, and ARM946E-S™ (Rev 1).

New operating system support

Windows Me.

Improved XScale processor support

Changes to the way XScale processors are supported means that it is no longer necessary to reset the processor when connecting to it. See *Intel XScale microarchitecture processors* on page D-3 for more information.

Improved hardware interface

The Multi-ICE interface unit now includes an external power supply option, enabling it to be connected to systems that cannot supply sufficient power.

Hot-plug support

The new power supply capabilities enable the Multi-ICE interface unit to be plugged into a running target board and the running program analyzed without restarting the target.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.5.4 Changes in Multi-ICE Version 2.1

Changes in Multi-ICE Version 2.1 are:

TAPOp API Minor changes to the TAPOp API clarify and streamline the affected functions.

1.5.5 New features in Multi-ICE Version 2.0

The new features in Multi-ICE Version 2.0 were:

New processor support

ARM7 cores ARM7TDMI-S™ (Rev 1 and Rev 2), ARM7TDI-S™ (Rev 2), and Samsung KS32C50100.

ARM9 cores ARM9E-S™ (Rev 0), ARM920T™ (Rev 1), ARM946E-S™ (Rev 0), and ARM966E-S™ (Rev 0).

ARM10 cores ARM1020T™ (Rev 0), ARM10200T™ (Rev 0).

Intel XScale microarchitecture

Intel 80200.

New operating system support

Windows 2000 and Solaris 7.0.

Support for the ARM Developer Suite

Full support for ADS v1.1, including the *Trace Debug Tools* (TDT).

Better target processor descriptions

Targets that have additional components, for example, system coprocessors and floating point units, can now be described and presented more clearly in the user interface, and are also no longer hard coded in the Multi-ICE DLL.

New Configuration dialog

The configuration dialog is now simpler to use, and includes support for the ARM TDT, and you can now search for Multi-ICE servers running on workstations using the Windows Network Browser service.

Embedded Trace Macrocell (ETM) and MultiTrace™ support

Multi-ICE supports access to the ETM.

MultiTrace is the ARM execution trace solution (available as a separate product) that includes the additional software and hardware.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RealMonitor support

Multi-ICE supports access to *RealMonitor* (RM), the ARM real-time debug solution.

EmbeddedICE/RT is supported

Multi-ICE supports processors containing ARM cores that contain the EmbeddedICE/RT extensions.

Autoconfiguration extensions

Autoconfiguration now supports the ARM Integrator™ boards in configuration mode, making the FPGA firmware on these boards easier to modify.

1.5.6 Changes in Multi-ICE Version 2.0

Changes in Multi-ICE Version 2.0 were:

Start Menu The Windows **Start → Programs** menu entry is now **ARM Multi-ICE v2.0** and the default install location is `Program Files\ARM`, for consistency with other ARM applications.

Breakpoint algorithm changes

The watchpoint and breakpoint allocation algorithms have been rewritten and so the actual allocation to hardware breakpoint units has changed.

For more detail see Appendix B *Breakpoint Selection Algorithm*.

`arm9_restart_code_address`

The ARM debugger internal variable `arm9_restart_code_address` has been removed, and an interface providing equivalent functionality added to the Multi-ICE configuration dialog.

1.5.7 New features in Release 1.4

The features that were new in Release 1.4 are:

New Processor Support

ARM920T (Rev 0) and ARM940T™ (Rev 1). A software update to Release 1.4 provided support for ARM966E-S (Rev 0).

New Operating System Support

Windows 98.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Support for the ARM Developer Suite

Full support for ADS v1.0, including support for RDI 1.5.1.

Improved demand paged memory support

Demand paged memory can make it hard to know where the system memory is in the address space, so extra facilities were added to support this.

Performance enhancements

Debugging performance was improved, especially with ARM9 cores.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 2
Getting Started

This chapter describes how to connect the parts of Multi-ICE together and how to configure the Multi-ICE server software. It contains the following sections:

- *System requirements* on page 2-2
- *Connecting the Multi-ICE hardware* on page 2-6
- *Connecting to nonstandard hardware* on page 2-11
- *Starting the software* on page 2-14.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.1 System requirements

This section describes the hardware and software requirements of Multi-ICE:

- *Host software requirements*
- *Host hardware requirements* on page 2-3
- *Target hardware requirements* on page 2-4.

2.1.1 Host software requirements

There are two distinct software components in Multi-ICE:

- the Multi-ICE server, that must be run on the computer the interface unit is attached to
- the Multi-ICE DLL, that can be run on another computer.

Table 2-1 identifies the operating systems you can use for each of these components.

Table 2-1 Supported operating systems for Multi-ICE

Operating system	Multi-ICE server	Multi-ICE DLL
Windows 95	Yes	Yes
Windows 98	Yes	Yes
Windows Me	Yes	Yes
Windows NT 4.0 (Intel)	Yes	Yes
Windows 2000	Yes	Yes
Solaris 2.6, 7.0, or 8.0	No	Yes
HP-UX 10 or 11	No	Yes
Redhat Linux 6.2 or 7.1	No	Yes

The graphical debuggers supplied in v1.0.1 (or later) of ADS, and SDT 2.51 are all compatible with the Multi-ICE DLL, as are debuggers supplied by third parties that conform to the ARM RDI 1.5.1 specification.

If you are running a debugger under UNIX, for example AXD, you must use another computer connected to it that can run the Multi-ICE server software. The workstation running UNIX must have networking software that supports a TCP/IP connection to the Multi-ICE server, and must meet the minimum software requirements specified in the debugger installation notes.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Networking software

If you require remote access to the server, your operating system must be installed with its supplied networking software. If the TCP/IP stack is not present during installation, the following warning text is displayed:

```
TCP/IP protocol does not appear to have been set up on this computer. Setup will
continue. Please install TCP/IP if you want to use Multi-ICE remote access
features.
```

Automatic dialup

Automatic dialup might be triggered when you use Multi-ICE because Multi-ICE uses network facilities. You can prevent unnecessary dialups by either:

- disabling **Allow Network Connections** on the Multi-ICE server **Settings** menu and only using This Computer as the server name in the DLL
- disabling automatic dialup.

2.1.2 Host hardware requirements

This section defines the minimum recommended hardware requirements for installing and running the Multi-ICE DLL and, on Windows, the Multi-ICE server.

Disk space

If you carry out a full installation of the software, up to 20MB of hard disk space is required.

To use the Multi-ICE software on Windows

To use the Multi-ICE server and DLL on Windows, you require the following:

- 200MHz Pentium processor
- system memory:
  - 32MB RAM for Windows 95, Windows 98, and Windows Me
  - 64MB RAM for Windows NT 4.0 and Windows 2000.
- CD-ROM drive (this can be a networked CD-ROM drive)
- an OS supported graphics device capable of VGA resolution or better
- parallel port
- network card (if remote access to the server is required).

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To use the Multi-ICE DLL on Solaris

To use the Multi-ICE DLL on Solaris, you require the following:

- Sun UltraSparc or compatible machine
- Solaris 2.6, 7.0, or 8.0, with the *Common Desktop Environment* (CDE)
- CD-ROM drive (this can be a networked CD-ROM drive)
- connected network interface.

To use the Multi-ICE DLL on HP-UX

To use the Multi-ICE DLL on HP-UX, you require the following:

- HP PA-RISC v1.1 or v2.0 processor
- HP-UX 10.20 or 11
- CD-ROM drive (this can be a networked CD-ROM drive)
- connected network interface.

To use the Multi-ICE DLL on Linux

To use the Multi-ICE DLL on Linux, you require the following:

- 200MHz Pentium processor
- Redhat Linux 6.2 or 7.1
- CD-ROM drive (this can be a networked CD-ROM drive)
- connected network interface.

2.1.3 Target hardware requirements

Multi-ICE has been designed to be very flexible, but it has the following target hardware requirements:

- A device interface conforming to the IEEE1149.1 (JTAG) specification.
- Electronic signals available to the interface, and within the limits of current and voltage specified in Chapter 6 *System Design Guidelines*.
- An IDC connector on the target board wired as described in *Multi-ICE JTAG interface connections* on page F-2, unless you are using the old 14-way EmbeddedICE connector or you construct a new cable.
- A maximum cable length between target board and Multi-ICE interface unit of 20cm, unless one or more of the modifications described in Chapter 6 *System Design Guidelines* is used.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- One or more ARM architecture CPUs containing supporting debug logic that is linked into a JTAG scan chain. This includes most ARM7 and ARM9 cores, the ARM1020T core, and Intel XScale microarchitecture processors. It does not include the StrongARM® processors.

A full list of supported processors, including full CPUs, is provided with the installation in the file `proclist.txt`.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.2 Connecting the Multi-ICE hardware

This section explains how to set up the hardware for Multi-ICE:

- *What you require*
- *Connection instructions* on page 2-8.

2.2.1 What you require

To set up the hardware you require the following items from the Multi-ICE product kit, shown in Figure 2-1 on page 2-7:

- the parallel cable, a round cable with a male D connector at each end
- the JTAG cable, a flat ribbon cable with a square *Insulation Displacement Connector* (IDC) socket at each end
- the Multi-ICE interface unit.

Depending on the type of target hardware you are using you might also require the following items:

- The supplied power adaptor, set to supply 9V to the Multi-ICE interface unit using a 2.1mm jack plug. Using this prevents Multi-ICE drawing its power from the target.
- The JTAG interface adaptor, ARM part number HPI-0027B. This is a small PCB with one 14-way and one 20-way connector mounted on it, and is available from ARM on request.

You must also provide the following items:

- a Windows workstation with an available parallel port, running an operating system supported by the Multi-ICE server (see Table 2-1 on page 2-2)
- some target hardware containing a JTAG-capable device supported by Multi-ICE (see *Target hardware requirements* on page 2-4).

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

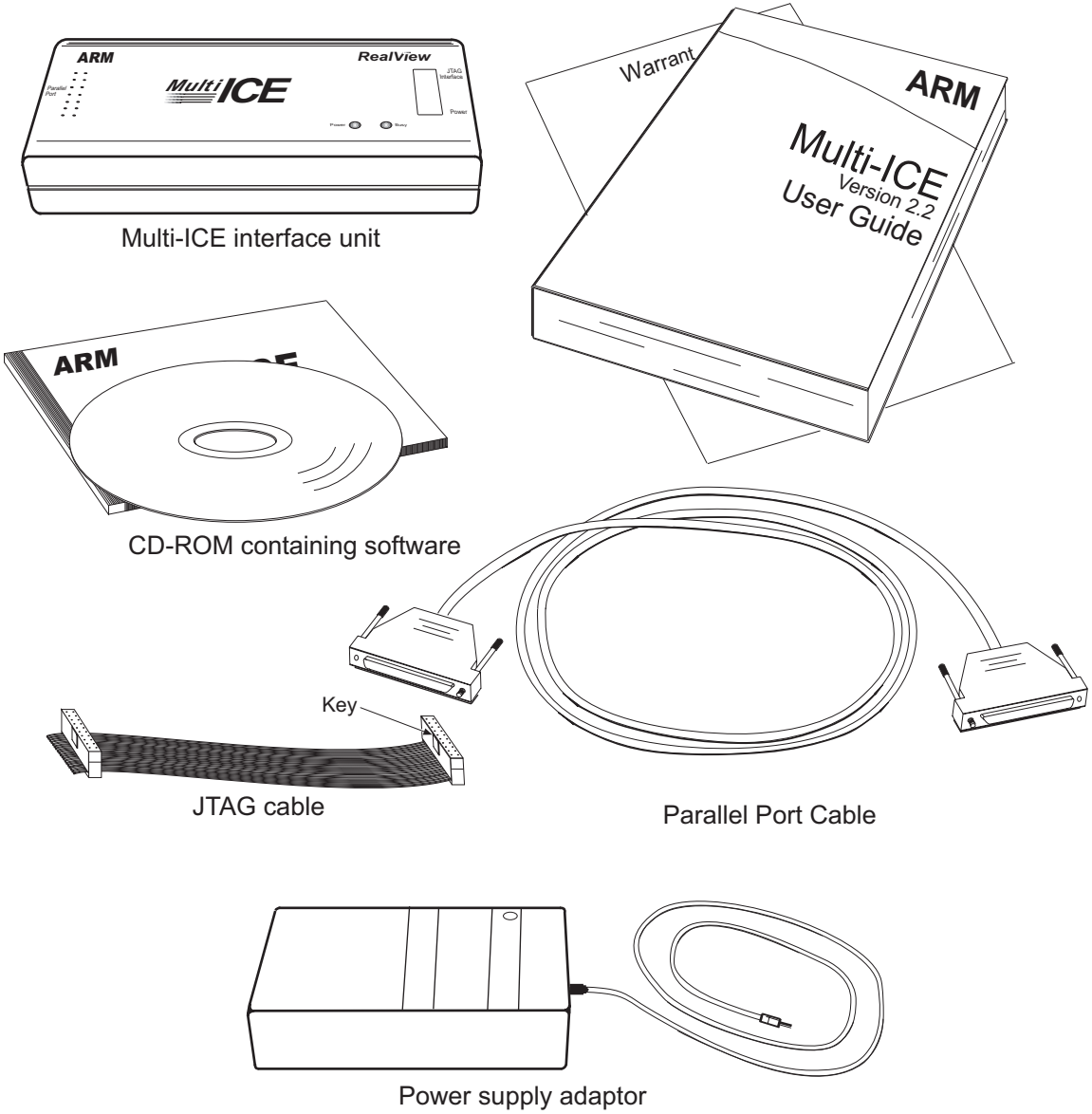


Figure 2-1 The Multi-ICE product kit

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.2.2 Connection instructions

You must connect the Multi-ICE interface unit to the your workstation and to the target hardware. An example is shown in Figure 2-2.

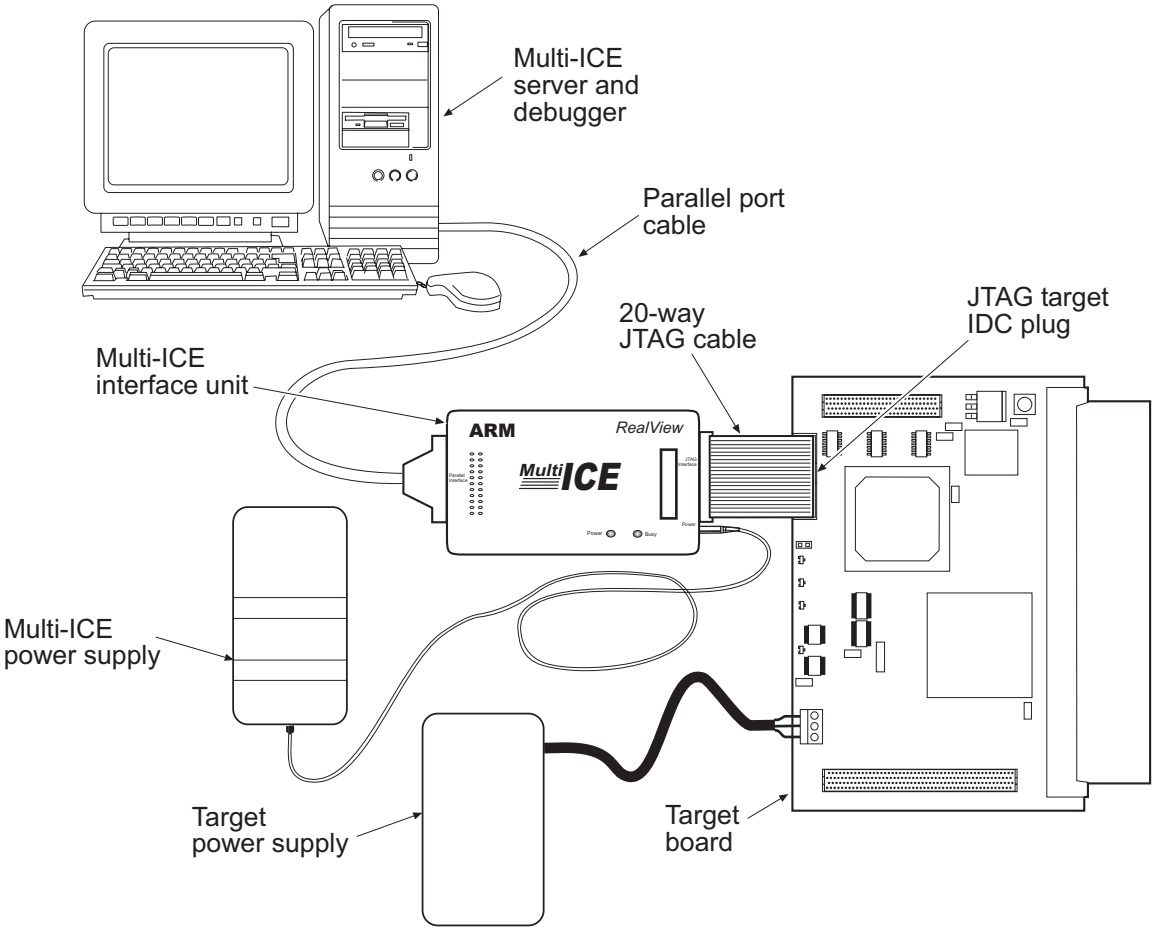


Figure 2-2 Multi-ICE interface unit cable connection

To connect the hardware together:

1. Ensure the Multi-ICE software is installed on the host machine. This is described in the section on installing the Multi-ICE software in the *Multi-ICE Installation Guide*.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. Connect one end of the parallel cable to the parallel port of the host computer (variously labeled *Printer*, *Parallel*, *IEEE 1284*, or with a graphic), and the other end of the cable to the Multi-ICE interface unit.

3. Ensure that the Multi-ICE unit has a power source:

   - Determine the Model No. of your Multi-ICE interface unit. This is printed on the underneath of the unit. If the interface unit is model 83 or later, and it is to draw power from the target, you must ensure that a link is fitted to jumper J8. This jumper is situated under the removable cover on the interface unit, as shown in Figure 2-3.

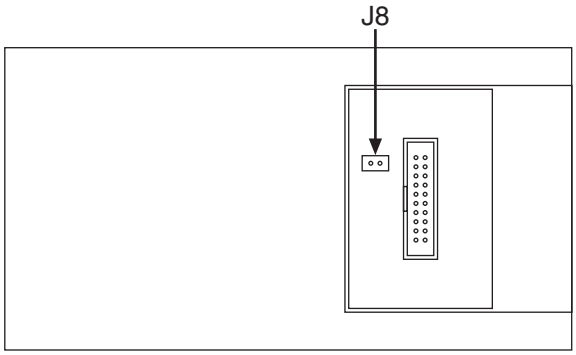


Figure 2-3 Location of jumper J8

   - If independent power to the Multi-ICE interface unit is required, for example because the target cannot supply sufficient current or because you are connecting to an already-running target, you must connect the external power supply to the Multi-ICE interface unit and switch it on.

     Note

     If you are connecting Multi-ICE to hardware that is powered and running, for example to find out what is happening, refer to *Post-mortem debugging* on page 4-45.

4. Connect one end of the JTAG cable to the JTAG connector on the Multi-ICE interface unit, and the other end of the cable to either:

   - the JTAG connector on the target board, if this is a 20-way IDC connector conforming to the Multi-ICE connection standard
   - the 20-way connector on the optional JTAG interface adaptor, if the target JTAG connector is a 14-way IDC connector conforming to the EmbeddedICE Interface Unit connection standard.

   The IDC sockets used for this cable are keyed using a small protrusion that must be matched up with a slot in the plug.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the target board has another variety of connector, see *Connecting to nonstandard hardware* on page 2-11.

5. If it is not already powered up, switch on the power to the target board.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.3 Connecting to nonstandard hardware

This section describes how to set up the Multi-ICE hardware when the target board does not have the ARM style 20-way IDC connector. It is split into the following sections:

- *Compatibility with PID, PIE, and PIV ARM development boards*
- *Nonstandard connectors*
- *Power supply* on page 2-12.

2.3.1 Compatibility with PID, PIE, and PIV ARM development boards

To use Multi-ICE with early ARM development boards, you must short the following resistors:

ARM7TDMI® header (HHI-0016B)

Resistor R1 (only if modification box number 1 is not marked with an X).

ARM PIV7T board (HBI-0008B)

Resistor R12.

ARM PIE7 board (HBI-0004B)

Resistor R53.

These modifications do not make the target board incompatible with the EmbeddedICE interface unit.

Note

You must manually reset the ARM Development Board (PID) before loading and running an image because power-up does not always provide a clean reset.

2.3.2 Nonstandard connectors

The Multi-ICE product is supplied with cables using 20-way IDC sockets wired to the ARM standard. Plugs suitable for this connector are fitted on all current ARM development boards and several third-party target boards. Some ARM development targets, for example the ARM Development Board (PID) CPU header cards, use a 14-way socket that is signal-compatible with the new 20-way socket. An adaptor card is available from ARM on request to allow connection to these boards.

Boards made using the *Texas Instruments* (TI) JTAG interface definition use the same 14-way IDC connector as the ARM boards, but use a different signal assignment. If you think your target might use this connector (for example, if the board is made by TI), you

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

must check the target board reference manual *before* using Multi-ICE. An adaptor to allow Multi-ICE to connect to these boards is available from ARM free of charge on request. Quote part number HPI 0068A.

If the target you are using does not use an ARM style connector, or you are involved in designing a target board, refer to the *Multi-ICE TAPOp API Reference Guide* for more information.

2.3.3 Power supply

Power is supplied to the Multi-ICE interface unit through:

- pin 2 (**Vsupply**) on the 20-way JTAG connector, drawing current from the target power supply

  Note

  On Multi-ICE Version 2.1 and later, this power passes through jumper J8.

- the power input jack.

The minimum target power supply voltage is 2V, and the maximum is 5V. You can calculate the approximate operating current using the formula:

$$800\text{mA} \times \left(\frac{1.5\text{V}}{targetV}\right)^2$$

A graph of this function is shown in Figure 2-4 on page 2-13. On power-up, the Multi-ICE interface unit draws more current than the graph shows, and the power supply must be capable of delivering this. As a general guide, 440mA at 3.3V has been measured. If the target supply voltage or its current capability is too low, you must use the external power input jack.

You can provide power to the Multi-ICE interface unit using the power input jack from a consumer power transformer (sometimes referred to as a wall-wart). The transformer must be rated to supply between 9 and 12V at 500mA minimum. If you use the external power jack, you can connect to targets using logic voltages of 1V to 5V.

Note

The original EmbeddedICE interface unit literature stated that the target power supply $\mathbf{V_{dd}}$ must have a series resistor in the feed to the JTAG interface power pin. For Multi-ICE to operate correctly from the target power supply, this resistor *must not* be present.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Shorting out this resistor does not affect the operation of the ARM EmbeddedICE Interface Unit, and, on target boards built by ARM Limited, does not affect the operation of the target board.

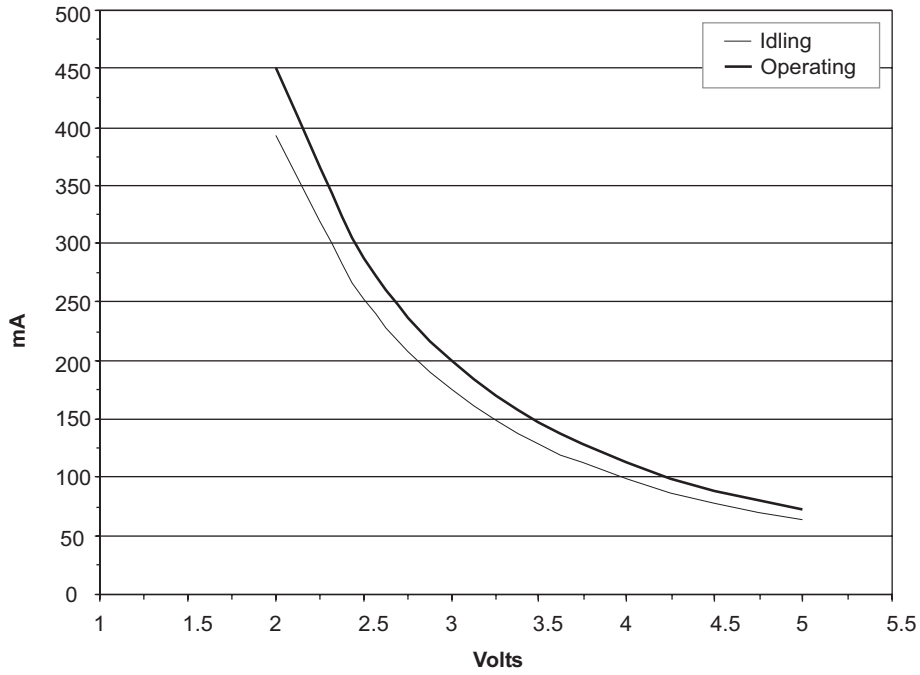


Figure 2-4 Multi-ICE current consumption with voltage

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.4 Starting the software

This section explains how to get the Multi-ICE software running. More detailed information on the software is provided in:

- Chapter 3 *Using the Multi-ICE Server*
- Chapter 4 *Debugging with Multi-ICE*.

If you have not already installed the software, do so now. Details on how to install the software are given in the *Multi-ICE Installation Guide*.

The following sections describe:

- *Microsoft Windows start program menu for Multi-ICE*
- *Starting the Multi-ICE server* on page 2-15
- *Other Multi-ICE server startup features* on page 2-17.

2.4.1 Microsoft Windows start program menu for Multi-ICE

After you have installed Multi-ICE on a Microsoft Windows computer, the menu items shown in Figure 2-5 are available on the Windows Programs menu. The order of items on this menu might differ from that shown.

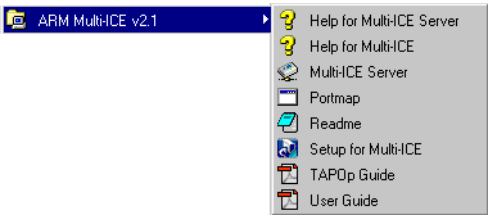


Figure 2-5 Start menu items for Multi-ICE

The items are:

Help for Multi-ICE Server

Displays the Multi-ICE server online help files.

Help for Multi-ICE

Displays the Multi-ICE online help files.

Multi-ICE Server

Runs the Multi-ICE server.

Portmap Runs the Portmap application, required for network access to the server.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Readme View the product `Readme` file in Windows Notepad. This contains any additional comments that are not included in the manuals.

Setup for Multi-ICE

Runs the Multi-ICE setup program. This enables you to install additional components, repair, or remove the Multi-ICE software from your workstation.

Files that have been moved from their original location, and files created in the Multi-ICE installation directory since the installation occurred, are not deleted when you remove or repair the installation.

TAPOp Guide

Display the Multi-ICE TAPOp API Reference Guide using the installed PDF file viewer.

Note

This item is only available if you have installed the PDF documentation.

User Guide Display the Multi-ICE User Guide using the installed PDF file viewer. The user guide is also supplied as a printed manual.

Note

This item is only available if you have installed the PDF documentation.

2.4.2 Starting the Multi-ICE server

To start the Multi-ICE server:

1. Ensure that:
   - the Multi-ICE interface unit is plugged into the workstation
   - the Multi-ICE interface unit is plugged into the target JTAG connector
   - the target is powered up
   - the green power light on the interface unit is glowing brightly.

   For more information refer to *Connecting the Multi-ICE hardware* on page 2-6.
2. Select **Start → Programs → ARM Multi-ICE v2.2 → Multi-ICE Server**.

   The software displays the Multi-ICE server window, shown in Figure 2-6 on page 2-16. The `portmap` application might also be started and minimized, depending on the host computer configuration.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

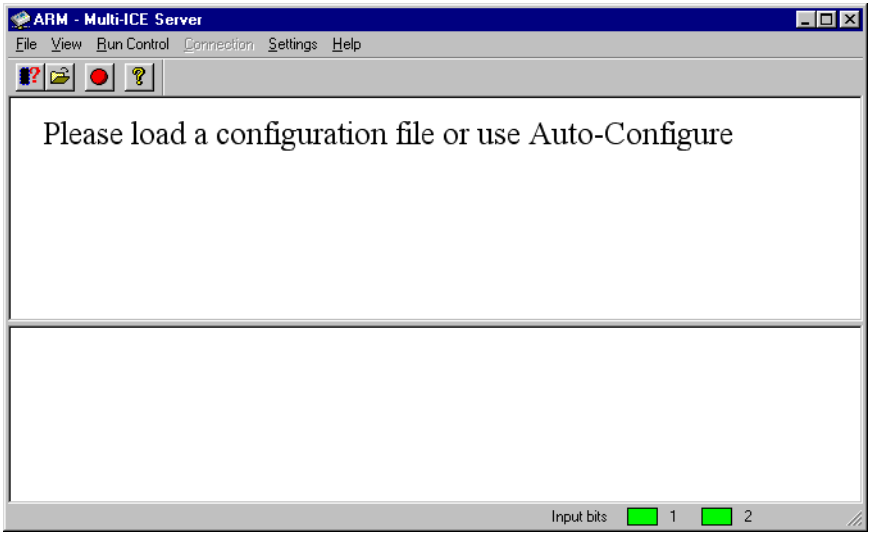


Figure 2-6 Unconfigured Multi-ICE server window

3. If a dialog box appears informing you that the Multi-ICE hardware cannot be found, click on OK and recheck the items listed in step 1.

4. Configure the server. This can usually be done using the Autoconfiguration command. Select **File → Auto-configure** and wait until the server has examined the target. If the server reports that the device is `UNKNOWN` then the server has to be configured manually. Refer to Chapter 3 *Using the Multi-ICE Server* for details of manually configuring the server. If the configuration works, the screen looks similar to Figure 2-7 on page 2-17.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

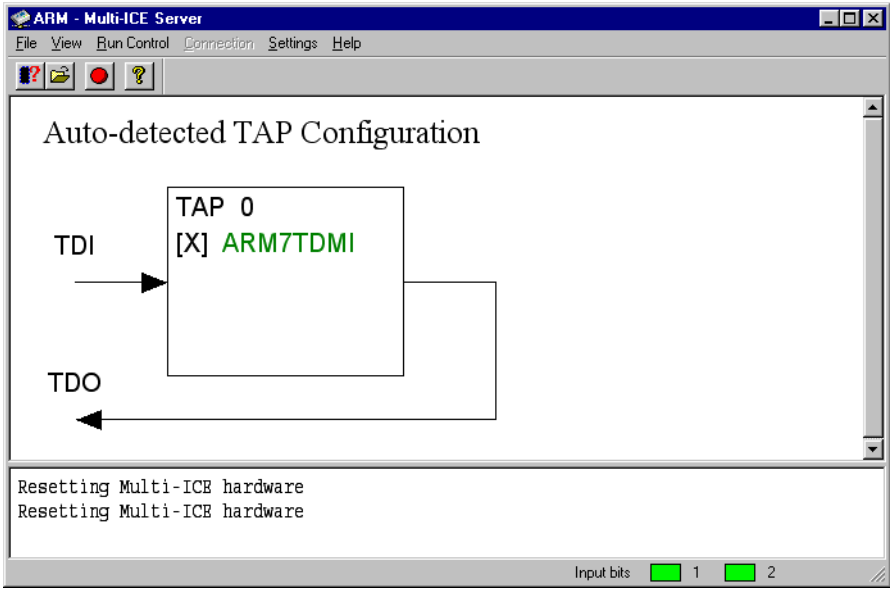


Figure 2-7 Multi-ICE server window configured for an ARM7TDMI

More information on configuration and the autoconfigure command is provided in *Server configuration* on page 3-14.

2.4.3 Other Multi-ICE server startup features

This section includes more information about starting the Multi-ICE server.

Using the server on a workstation without a TCP/IP stack installed

If there is no TCP/IP stack installed on the workstation when you start the Multi-ICE server for the first time, a warning message is displayed and the **Settings** item **Allow Network Connections** is automatically unchecked.

Network settings are remembered between sessions.

Starting without hardware

You can start the Multi-ICE server software without the interface unit being connected. A message box appears containing the text:

```
Could not find the Multi-ICE hardware. Please check that the hardware is properly
connected to the parallel port and powered up
```


Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This message is just a warning. To start using the server:

1. Ensure the interface unit is connected to the parallel port and to the target, and that it is powered.
2. Load a configuration into the server using **File → Load configuration** or **File → Auto-configure**.

No network connection

In some circumstances, on machines with no network software installed or with incorrect network settings, the Multi-ICE server terminates immediately after it starts up. This means you have no opportunity to switch off the **Allow Network Connections** setting (see *Using the server on a workstation without a TCP/IP stack installed* on page 2-17 and *Settings menu* on page 3-7).

If you experience this problem, run the script `Non_tcp_ip.reg` (in the Multi-ICE installation directory) by **Open**ing it in Windows Explorer. This prevents the server trying to use the network by switching off the **Allow Network Connections** setting in the system registry.

Starting the server minimized

You can create a shortcut icon to start the server minimized, as follows:

1. Right-click over the server icon and choose **Create Shortcut**.
2. Right-click over the shortcut and choose **Properties**.
3. Click on the **Shortcut** tab.
4. Select **Minimized** from the **Run** drop-down menu.
5. Click **OK**.

To start the server minimized, double click on the shortcut icon.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 3
Using the Multi-ICE Server

This chapter describes how you use the Multi-ICE server. It contains the following sections:

- *About the Multi-ICE server menus* on page 3-2
- *Multi-ICE server device configuration files* on page 3-9
- *Server configuration* on page 3-14
- *Using the Multi-ICE server with multiple processors* on page 3-25.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.1 About the Multi-ICE server menus

This section gives an overview of the menu items on the Multi-ICE server:

- *Menu structure*
- *File menu*
- *View menu* on page 3-5
- *Run control menu* on page 3-5
- *Connection menu* on page 3-6
- *Settings menu* on page 3-7
- *Help menu* on page 3-8.

3.1.1 Menu structure

Figure 3-1 shows the submenus and their items.

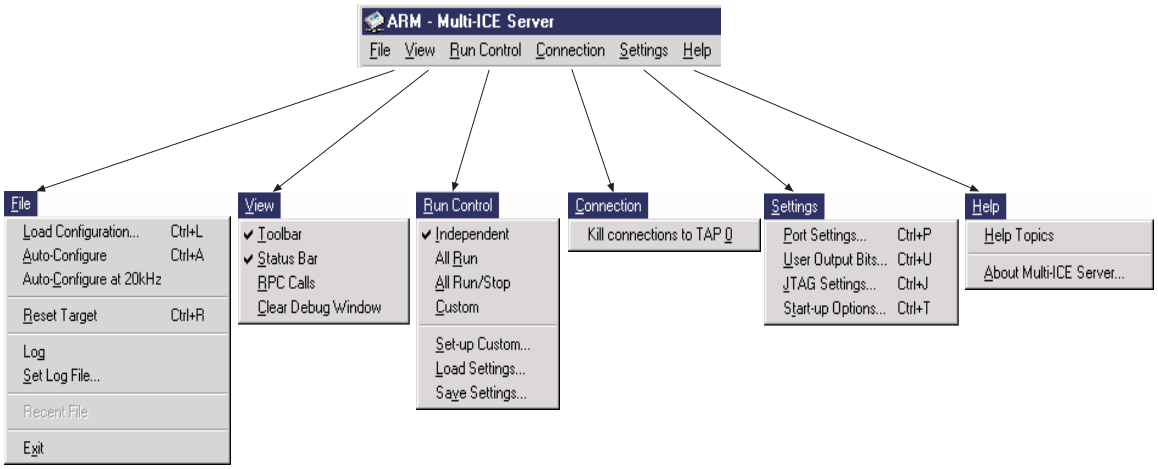


Figure 3-1 Multi-ICE server menu items

3.1.2 File menu

The File menu allows you to configure the Multi-ICE server and control the logging of TAPOp requests, and is shown in Figure 3-2 on page 3-3.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

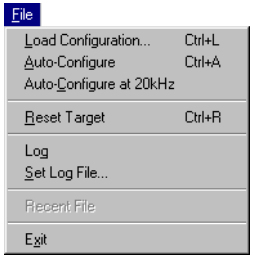


Figure 3-2 The File menu

The menu contains the following items:

Load Configuration

Displays a dialog box that you use to enter the name and path of a configuration file. This item is used for the manual configuration of Multi-ICE. This is described in *The IRlength.arm configuration file* on page 3-13.

Auto-Configure

Interrogates the device(s) connected to the JTAG scan chain and creates a configuration file containing what was found. See Table 3-1 for the actual frequencies used.

Table 3-1 TCK frequency for autoconfigure

Menu entry	TCK frequency during autoconfigure	TCK frequency in normal operation
Auto-Configure	1MHz	10MHz[a]
Auto-Configure at 20kHz	20kHz	20kHz

a. 1MHz for targets with more than one TAP, or for devices that are known to need a slower frequency unless adaptive clocking is used.

Autoconfiguration is described in *Automatic device configuration* on page 3-9. Unrecognized devices are marked as `UNKNOWN`, but you can add these to a lookup table, as described in *The IRlength.arm configuration file* on page 3-13.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

If autoconfiguration of a known processor fails (showing `UNKNOWN`), reset the processor using a hardware reset button or a power-cycle and try the autoconfigure again.

Auto-Configure at 20kHz

This item does the same as the `Auto-Configure` item, but uses a **TCK** frequency that never exceeds 20kHz. See Table 3-1 on page 3-3 for the actual frequencies used. This can be useful when the JTAG cable or the device is not capable of reliable operation at higher frequencies, or when the device might be in a sleep mode (when the slow system clock prevents the device responding sufficiently quickly to faster **TCK**s).

When manual configuration is used, the **TCK** frequencies indicated for **Auto-Configure** in Table 3-1 on page 3-3 are used unless the configuration file specifies alternate timing parameters.

Reset Target

Resets the target hardware. Clicking the **Reset Target** button in the toolbar is equivalent to selecting this menu item.

You can control the action of **Reset Target**, so that it asserts **nSRST**, **nTRST**, or both signals. (**nSRST** and **nTRST** are explained in the *Glossary*.) You can do this either from the JTAG **Settings** dialog (see *The JTAG Settings dialog* on page 3-21) or from a **Reset** section in the configuration file (see *Multi-ICE server device configuration files* on page 3-9).

You can also reset the target hardware from the debugger, using the `system_reset` internal variable (see *Debugger internal variables* on page 4-36). This method asserts **nSRST**, but does not assert **nTRST**.

Log Turns remote procedure call logging on or off. When turned on, a tick is displayed next to the menu item and text describing the TAPOp protocol requests received by the server are written to a log file.

Use **Set Log File** to specify the filename that is used.

Set Log File...

Displays a dialog box that you use to enter the name and path of a log file.

Recent File Displays a list of the eight most recent configuration files you have used.

Exit Closes the Multi-ICE server.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.1.3 View menu

The **View** menu controls the display of the Multi-ICE server window and is shown in Figure 3-3:

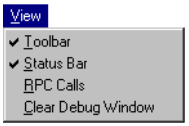


Figure 3-3 The View menu

The menu contains the following items:

Toolbar Turns the tool bar on or off. When the tool bar is displayed, a tick is displayed next to the menu item. The tool bar gives you quick access to the following functions:

File → Auto-Configure attempts to automatically configure the server

File → Load Configuration prompts you for the name of a configuration file

File → Reset Target resets the target using **nTRST** or **nSRST** or both

Help → Help Topics displays the Multi-ICE online help.

Status Bar Turns the status bar on or off. When the status bar is displayed, a tick is displayed next to the menu item. The **Status** bar displays information on the current state of Multi-ICE.

RPC Calls Turns the debug window on or off. When the debug window is active, a tick is displayed next to the menu item and all the TAPOp requests are displayed in the debug window.

Clear Debug Window

Clears all text in the debug window.

3.1.4 Run control menu

The **Run Control** menu controls the synchronized stopping and starting of multiple cores and is shown in Figure 3-4 on page 3-6.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

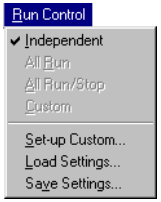


Figure 3-4 The Run Control menu

The menu contains the following items:

Independent Makes all configured devices behave independently. There is no interaction between devices.

All Run Starts all devices as close to simultaneously as the hardware allows.

All Run/Stop Makes all configured devices:

- run when all connected debuggers have indicated that their processor can start
- stop if any one of the devices stop.

Custom Makes configured devices interact as you have specified using the **Set-up Custom** action.

Set-up Custom... Displays a dialog box that enables you to specify the way in which devices interact. See *Setting up interaction between devices* on page 3-28 and *Setting up the poll frequency* on page 3-30 for more information.

Load Settings... Displays a dialog box that enables you to load run control settings previously saved using **Save Settings**.

Save Settings... Displays a dialog box that enables you to save the run control settings to a file.

The items on this menu are described in detail in the section *Using the Multi-ICE server with multiple processors* on page 3-25.

3.1.5 Connection menu

The **Connection** menu lists all TAP controllers in use and gives you the option to kill connections individually, and is shown in Figure 3-5 on page 3-7.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Figure 3-5 The Connection menu

Caution

Only kill connections when other attempts to disconnect the client have failed.

Killing a connection between the server and an active client can result in the client crashing or exhibiting other undesirable behavior.

3.1.6 Settings menu

The **Settings** menu allows you to configure the way the server uses the JTAG port, and is shown in Figure 3-6.

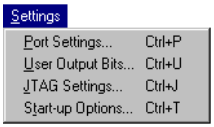


Figure 3-6 The Settings menu

The menu contains the following items:

Port Settings....

Displays a dialog box that you use to select the required parallel port address, with the option of forcing 4-bit access. It also shows the current port mode. Port settings are described in more detail in *Parallel port settings dialog* on page 3-18.

User Output Bits...

Displays a dialog box to control the user output bits. These bits correspond to two logic-level outputs available from the User *Input/Output* (I/O) connector (see *User output bits dialog* on page 3-19 and the *Multi-ICE TAPOp API Reference Guide*). You can use these signals to remotely control user logic at the server location.

JTAG Settings...

Displays a dialog box that you use to set the clock speed. You can select the clock speed from preset frequencies, by using the **Set Periods Manually** option, or by including the information in a configuration file.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

See section *JTAG settings dialog* on page 3-21 for more information. If timing information is included in the configuration file, it is selected automatically.

Start-up Options

Displays a dialog box that you use to specify what the server does when it starts up. For more information, see *Start-up Options dialog* on page 3-16.

3.1.7 Help menu

The **Help** menu gives you access to the online help and some information on Multi-ICE, and is shown in Figure 3-7.

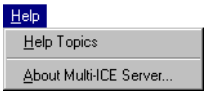


Figure 3-7 The Help menu

The menu contains the following items:

Help Topics Starts Multi-ICE help.

About Multi-ICE Server...

Displays version information for software and hardware modules of Multi-ICE. You must provide this information when contacting your vendor technical support.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2 Multi-ICE server device configuration files

The Multi-ICE server uses a configuration file to store information on the devices on the board. Multi-ICE requires configuration data to select the correct driver. It already has configuration data on the supported ARM devices.

Creating configuration files is described in the following sections:

- *Automatic device configuration*
- *Manual device configuration* on page 3-12.

3.2.1 Automatic device configuration

If all cores in your ASIC are supported ARM cores, Multi-ICE can create the configuration file by scanning the ASIC and creating the file `autoconfig.cfg`. Do this using the menu item **File → Auto-Configure**, or for slower devices use **File → Auto-Configure at 20KHz**.

The configuration file contains information on each TAP controller. The Multi-ICE product contains the necessary information on all supported JTAG capable ARM devices. If non-ARM devices are used in the device, you can declare these to Multi-ICE so that they are named in the configuration diagram.

To create a configuration file automatically, select **File → Auto-Configure**. Multi-ICE displays a pictorial representation of the devices found and the order in which they appear in the scan chain. Figure 3-8 on page 3-10 shows the result of configuring an ARM940T.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

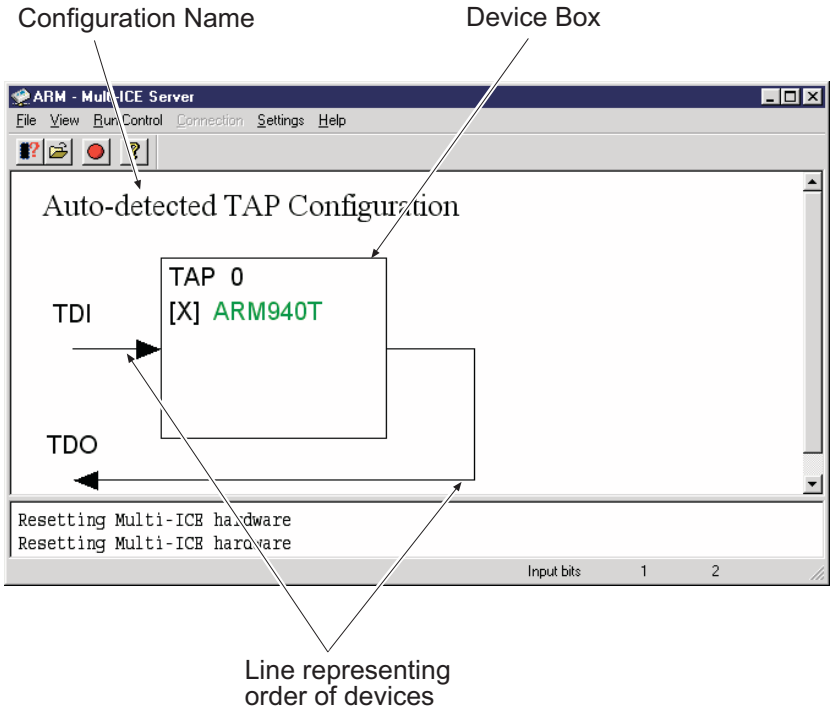


Figure 3-8 Autoconfiguring an ARM940T

Note

Autoconfigure issues several system resets while determining the configuration. If this must not be done for your target, you must configure the server manually. Refer to *The IRlength.arm configuration file* on page 3-13.

Figure 3-8 shows:

- The name of this configuration. This enables you to label your configurations.
- The order of the devices found in the scan chain. The arrow marked **TDI** (Test Data In) represents JTAG data coming from the Multi-ICE interface unit, and the arrow marked **TDO** (Test Data Out) represents data going into the Multi-ICE interface unit.
- A box for each device found. Within the box are:
  - The state of each core using the following symbols:
    [S] denotes that the core is stopped

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[R] denotes that the core is running

[D] denotes that the core is downloading

[X] denotes that the core state is unknown (no debugger connected).

— The TAP number of this device.

— The generic name of this device. This name is the name listed in `irlength.arm` and cannot be changed.

Multi-ICE writes the automatically generated configuration to a file as well as loading it for immediate use. The file is written to the Multi-ICE program directory (for example, `C:\Program Files\ARM\ARM Multi-ICE\`) with the name `autoconf.cfg`. The file corresponding to the configuration shown in Figure 3-8 on page 3-10 is shown in Example 3-1.

Example 3-1 Autoconfig file for an ARM940T

```
;Total IR length = 4
[TITLE]
Auto-detected TAP Configuration
[TAP 0]  ;IR_len=4, ID_code=1F0F0F0F
ARM940T
[Timing]
Adaptive=OFF
```

The file contains the configuration name, an entry for TAP 0 that references the ARM940T, and some timing data. The semicolon ; introduces a comment into the file that continues to the end of the line.

For a detailed description of the contents of this file, see Appendix A *Server Configuration File Syntax*.

Note

By copying and renaming the automatically generated configuration file you can create a number of different server configurations without having to write your own configuration file or let Multi-ICE reset the board as part of an autoconfigure.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2.2 Manual device configuration

You configure the Multi-ICE server manually by creating a server configuration file and loading it into the Multi-ICE server. A complete description of the structure and contents of the configuration file is provided in *Device configuration file* on page A-3. To create the file:

1. Start by checking `proclist.txt` to determine if the device you are configuring is supported by Multi-ICE:
   - If the device type is not supported, you can name the device by creating a device entry that tells Multi-ICE how long the scan chain select register for the device is. See *Naming an unsupported device* on page 3-13.
   - If the device type is supported, note the name of the device entry in the file `IRlength.arm`. See *The IRlength.arm configuration file* on page 3-13 for more information about this file.
2. Open a text file editor, and create a new file, using the file extension `.cfg`.
3. Enter entries in the text file for at least:
   - A `TITLE` section.
   - A `TAP 0` section, containing the name used in `IRlength.arm` for the first device.

   Additional sections in the file might be required depending on the devices you are configuring. Refer to *Device configuration file* on page A-3 for more information.
4. Save the file.
5. In the Multi-ICE server, click **File → Load Configuration...**.
6. Locate the configuration file you created, and click **Open**.
7. The Multi-ICE server is configured to expect the devices you put in the configuration.

   Note

   When you load a manual configuration, the Multi-ICE server does not check that the devices you configure match the actual scan chain. It is your responsibility to ensure this.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Naming an unsupported device

To name a single UNKNOWN device:

1. You must find out the length of the TAP controller instruction register (IR) in the unknown device. You can do this by:
   a. Autoconfiguring the target.
   b. Double clicking on the UNKNOWN device box in the Multi-ICE server window. A dialog box appears that includes the IR length.
2. Create a text file called USERDRV*n*.TXT (where *n* is the IR length of the unknown device) to the Multi-ICE installation directory. The file must contain the name of the device.

For example, to name a target that included a TAP controller for a DSP that had an IR length of 4 bits, you would create a file called USERDRV4.TXT containing the text DSP.

Note

You cannot name multiple devices that have the same IR length because the Multi-ICE server cannot distinguish between them.

The IRlength.arm configuration file

The Multi-ICE server calculates the length of the scan chain by adding the length of each IR register it finds in the JTAG chain. This information is held in a file called IRlength.arm. You can edit this to store information on other devices that you use, for example a DSP. This file is stored in the Multi-ICE installation directory. An extract from the IRlength.arm file is shown in Example 3-2.

Example 3-2 Extract from irlength.arm configuration file

```
;ARM7 series cores
ARM7TDMI=4
ARM7TDMI-S=4
ARM740T=4
;ARM9 series cores
ARM9TDMI=4
ARM920T=4
```


Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.3 Server configuration

This section describes in more detail the configuration of the Multi-ICE server program. In particular, it describes:

- *Chip driver settings dialog*
- *Start-up Options dialog* on page 3-16
- *Parallel port settings dialog* on page 3-18
- *User output bits dialog* on page 3-19
- *User input bits* on page 3-21
- *JTAG settings dialog* on page 3-21.

3.3.1 Chip driver settings dialog

If you double-click on the square outline of the core in Figure 3-8 on page 3-10 the Multi-ICE server provides more information about the devices it is connected to. The device information is obtained by reading the device TAP controller settings and interpreting them according to a standard table that is built into the server. The dialog box is shown in Figure 3-9 on page 3-15.

At the top of the dialog box the **List of Drivers** contains the names of the Multi-ICE drivers that are applicable to a particular TAP controller. The list always contains the main driver (the ARM940T in Figure 3-9 on page 3-15), which has a positive IR Length. It might also contain alias drivers, declared in the `irlength.arm` file with an IR Length of zero. An example of this is an ARM920T with an ETM attached.

Information in the **Driver Details** region of the dialog box changes for every connection, and so also between each device in the **List of Drivers**:

Connected To The name assigned to the connection.

At The time this connection was made.

connectId The connection number assigned to the connection, which is also visible in the RPC Log file.

Vers. Reqd The version number of the server protocol that the client requested when the connection was made.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

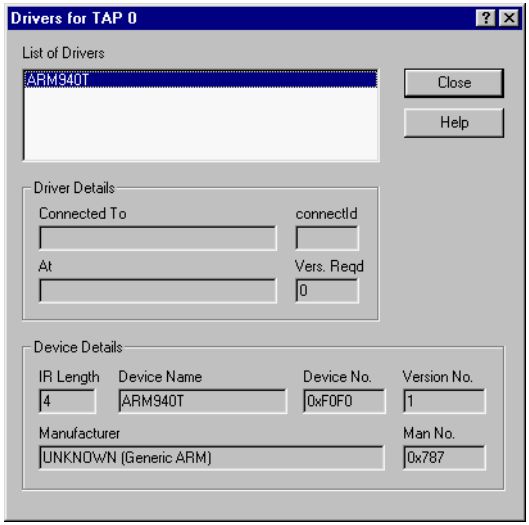


Figure 3-9 TAP driver status dialog

Information about the device itself is available in the **Device Details** region of the dialog box. Other than the **Device Name**, the information in this region does not change when a different driver is selected in the **List of Drivers** because it relates to the device, not the driver used.

The dialog box contains the following items:

IR Length The length in bits of the Instruction Register, a primary element in the JTAG TAP Controller.

Device Name The name given to the device.

Device No. The number assigned by the manufacturer of the device.

Version No. The chip version number.

Manufacturer A textual version of the manufacturer number.

Man No. The JTAG manufacturer code.

The device, version, and manufacturer numbers are read from the ID register in the TAP controller. Values in this register conform to the format shown in Figure 3-10 on page 3-16.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

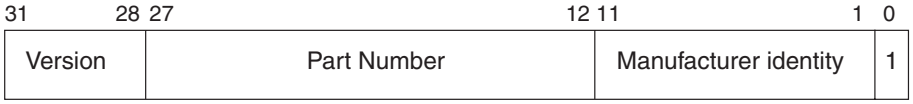


Figure 3-10 TAP Controller Device ID register format

Note

According to the IEEE 1149.1 specification for device identifiers the value `0xF0F0` in the **Device No.** field and `0x787` in the **Man No.** field is invalid.

ARM Limited puts these values into the device logic when the logic is supplied to the silicon manufacturer, with the intention that it is overridden with manufacturer and device-specific codes.

If the default ARM values are not overridden, the Multi-ICE DLL uses the string `UNKNOWN (Generic ARM)` as the manufacturer. The device name and part number are determined from the device characteristics when this is possible. When it is not possible the device fails to autoconfigure and must be manually configured. Multi-ICE also displays the true manufacturer number and string if this can be determined.

3.3.2 Start-up Options dialog

This section describes the parts of the Start-up Options dialog shown in Figure 3-11.

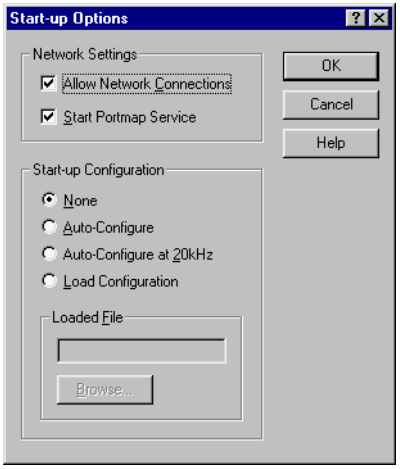


Figure 3-11 The Start-up Options dialog

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The dialog box is accessed from the **Settings → Start-up Options...** menu, and is split into two groups:

- **Network Settings**
- **Start-up Configuration**.

The **Network Settings** items enable or disable the network abilities of the Multi-ICE server. Other settings control the initial JTAG device configuration.

The dialog box contains the following items:

Allow Network Connections

When selected, the Multi-ICE server enables its clients to connect remotely over a network using the Sun RPC protocol. This requires that a TCP/IP stack is set up on both workstations. However, you can deselect it to:

- prohibit network connections to sensitive devices
- enable a workstation without a TCP/IP stack set up to use the server.

Start Portmap Service

If you select this option, when you next start the Multi-ICE server the Portmap program also starts. If the **Start Portmap Service** box is not checked, you must start the portmap service in another way (for example, by selecting **Start → Programs → ARM Multi-ICE v2.2 → Portmap**) before the Multi-ICE server is started.

Note

If the automatic start of the portmap service is not selected and a portmapping service is not being provided by another application, no warning message is displayed. The result is that the client fails to connect to the server and after a short time it reports that an error has occurred.

To overcome this you must start the Multi-ICE server again, select the **Start Portmap Service** option, close the Multi-ICE server, and then restart the Multi-ICE server.

This option is disabled when **Allow Network Connections** is deselected, because the portmapping service is not required.

None If this is chosen, no autoconfiguration or load configuration file action is performed. When the server starts up the window looks similar to Figure 2-6 on page 2-16 and the server must be configured before it can be used.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Auto-Configure

This option automatically creates a configuration file naming all devices found as described in *Automatic device configuration* on page 3-9. Unrecognized devices are marked as `UNKNOWN`.

Auto-Configure at 20kHz

The 20kHz autoconfigure is useful with systems that have slow clocks, such as emulation environments, or with processors in Sleep mode.

Load Configuration

Presents a dialog box for you to enter the name and path of a configuration file. This option is used for the manual configuration of Multi-ICE, and is described in *The IRlength.arm configuration file* on page 3-13.

3.3.3 Parallel port settings dialog

This section describes how you use the **Port Settings** dialog. This is accessed from the **Settings** menu.

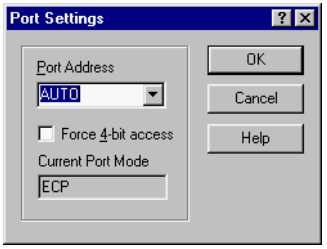


Figure 3-12 The Port Settings dialog

The dialog box contains the following items:

Port Address

Is the parallel port address to be used. Can be one of:

AUTO Automatically selects the parallel port to use.

LPT1 Selects LPT1 as the parallel port to use.

LPT2 Selects LPT2 as the parallel port to use.

Note

You must connect the Multi-ICE interface to a parallel port that uses standard parallel port hardware at addresses `0x278` or `0x378`.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Force 4-bit access

Forces the parallel port to use 4-bit data transfer.

Current Port Mode

This item indicates the operational mode of the parallel port. Many parallel ports can operate in one of several modes, defined by the BIOS settings. See your computer hardware manual for more information. Multi-ICE understands the following parallel port modes:

4-bit Basic unidirectional parallel port.

8-bit 8-bit bidirectional parallel port.

ECP Enhanced Capability Port. This mode is faster than 8-bit mode because it is possible to use block data transfers.

Multi-ICE does not understand the *Enhanced Parallel Port* (EPP) mode, and uses these ports in bidirectional 8-bit mode. Multi-ICE can use an IEEE 1284 port in ECP compatibility mode.

Note

- The Windows 95, Windows 98, and Windows Me drivers do not use ECP mode because the ECP-aware parallel port driver interferes with the operation of the Multi-ICE parallel port driver.
- Some machines might exhibit random communication failures due to nonconforming parallel ports. If you have difficulty in connecting to the Multi-ICE hardware, or experience timeout errors during operation, try forcing 4-bit access.

3.3.4 User output bits dialog

The user output bits correspond to two TTL logic-level outputs available from the user I/O connector (see Appendix G *User I/O Connections*). You can use these signals to remotely control user logic at the server location.

You access the **User Output Bits** dialog shown in Figure 3-13 on page 3-20 from the **Settings** menu.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

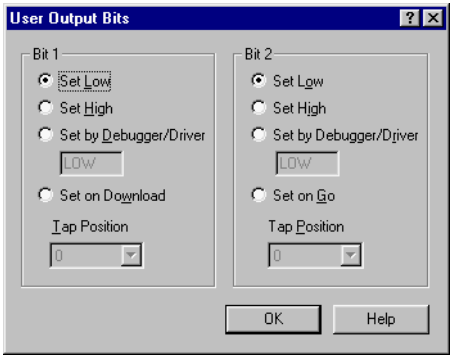


Figure 3-13 The User Output Bits dialog

Note

The state of the user output bits changes as soon as you click on them. You do not have to click on **OK** for the changes to take effect.

The dialog box contains the following items:

Set Low Turns the bit permanently LOW.

Set High Turns the bit permanently HIGH.

Set by Debugger/Driver

Enables the output bits to be controlled by either:

- the TAPOp procedure calls `TAPOp_WriteMICEUser1` and `TAPOp_WriteMICEUser2`, described in the *Multi-ICE TAPOp API Reference Guide*
- the debugger internal variables `output_bit_1` and `output_bit_2`, described in *Internal variable descriptions* on page 4-39.

Set on Download

Sets bit 1 HIGH while a debugger is downloading to the specified TAP controller.

Set on Go Sets bit 2 HIGH while a debugger is executing an image file on the specified TAP controller.

You can select the TAP controller to use with the user output bits by selecting its number from the **Tap Position** drop-down box. All configured TAP controllers are listed.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.3.5 User input bits

The user input bits correspond to two TTL logic-level outputs available from the user input/output connector (see Appendix G *User I/O Connections*). You can use these signals to remotely control user logic at the server location.

The user input bits are shown at the bottom-right corner of the Multi-ICE server window as shown in Figure 3-14. Each bit is:

- light green when HIGH
- dark green when LOW.



Figure 3-14 Status of the user input bits

3.3.6 JTAG settings dialog

The JTAG Settings dialog shown in Figure 3-15 enables you to select how Multi-ICE generates the JTAG clock and reset signals. The menu entry is disabled until the target is configured. You must either autoconfigure the target or load a configuration file.

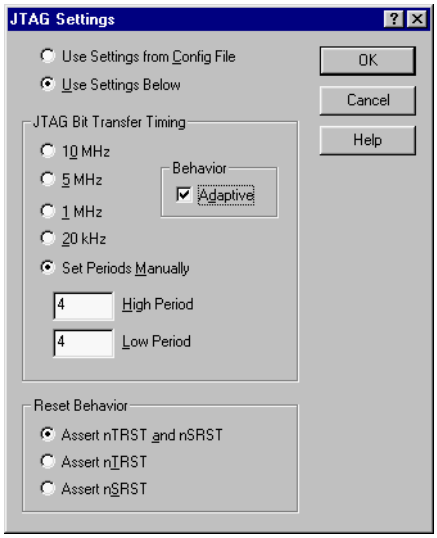


Figure 3-15 The JTAG Settings dialog

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Settings that are not defined in the selected target configuration file are set to a default value. Using the dialog box you can:

- use timing and reset signal settings from the selected configuration file
- define your own timing and reset signal settings.

To use settings defined in the configuration file unchanged, click **Use Settings from Config File**.

To modify the settings defined in the configuration file, click **Use Settings Below**. Doing this enables the **JTAG Bit Transfer Timing** and **Reset Behavior** control groups. The initial setting for these controls is taken from the current configuration of the server.

In the **JTAG Bit Transfer Timing** group, you can define the basic **TCK** frequency, and choose whether adaptive clocking is used:

- Select a preset **TCK** frequency from: **10MHz**, **5MHz**, **1MHz** or **20kHz**.
- Select **Set Periods Manually** to define a specific **TCK** pattern. Refer to *Setting TCK periods manually*.
- Select **Adaptive** when adaptive clocking must be used for the target device. Refer to *Adaptive clocking* on page 3-24 and Chapter 6 *System Design Guidelines*.

In the **Reset Behavior** group, you can define the signals that are asserted when you tell Multi-ICE to reset the target system. There are two signals that the Multi-ICE interface unit can control:

nTRST When connected correctly, asserting this signal resets only the JTAG logic on the target.

nSRST When connected correctly, asserting this signal resets the target processor and connected peripherals, but does not reset the JTAG logic.

The radio buttons enable you to select the combination that is asserted when you choose **File → Reset Target** or click on the red reset toolbar button.

You must refer to the target documentation to determine the actions these signals have for your target.

Setting TCK periods manually

If you select **Set Periods Manually**, or include the clocking information in a configuration file, the periods and frequency are calculated using the following formulas:

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
HIGH period  =  50ns * (high_scale * (high_multiplier + 1))
LOW period   =  50ns * (low_scale * (low_multiplier + 1))
Total period = HIGH period + LOW period

Frequency    = 1 / Total period
```

For a precalculated table of frequencies and values, refer to *TCK frequencies* on page F-7 and *TCK values* on page F-11. The same HIGH and LOW period values are used in the server configuration file (see Appendix A *Server Configuration File Syntax*)

Note

At very low JTAG clock rates, the parallel port driver used by the server uses a large proportion of the workstation processing time. This causes any applications that are running to execute at a reduced speed.

You can enter values between 0–255 for the HIGH and LOW periods. The 8-bit values you enter are split into three and five bits to form the scale (`S`) and the multiplier (`M`) values as shown in Table 3-2.

Table 3-2 Scale and multiplier values

Scale			Multiplier				
7	6	5	4	3	2	1	0
S	S	S	M	M	M	M	M

The multiplier is formed from the lower five bits, allowing values from 0 to 31. Table 3-3 shows the scale value. *SSS* are the three most significant bits. *Scale* is the value to be used in the formula:

Table 3-3 Scale values for clocking speeds

SSS	Scale
0	1
1	2
2	4
3	8
4	16

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-3 Scale values for clocking speeds (continued)

SSS	Scale
5	32
6	64
7	128

Example 3-3 shows how you might encode some sample frequencies:

Example 3-3 Deriving values for JTAG HIGH and LOW settings

```
100kHz (approx)   HIGH = LOW = 162  [SSS = 5 (S = 32)       M = 2]
500kHz            HIGH = LOW = 19   [SSS = 0 (S = 1)        M = 19]
2MHz              HIGH = LOW = 4    [SSS = 0 (S = 1)        M = 4]
```

Adaptive clocking

If the target provides the **RTCK** signal, select the **Adaptive clocking** function to synchronize the clock to the processor clock outside the core. This ensures there are no synchronization problems over the JTAG interface.

Note

If you use the adaptive clocking feature, transmission delays, gate delays, and synchronization requirements result in a lower maximum clock frequency than with non-adaptive clocking. Do not use adaptive clocking unless it is required by the hardware design.

For a full description of the concept of adaptive clocking, see Chapter 6 *System Design Guidelines*.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4 Using the Multi-ICE server with multiple processors

If you have more than one processor in your target, Multi-ICE enables you to connect a separate debugger to each of your processors. You can debug code running on each processor entirely independently, setting breakpoints, downloading images, or start and stop processors, without affecting the other processors.

To do this:

1. Ensure the Multi-ICE server is showing all processors on the target.
2. Run an instance of your chosen debugger for each processor.
3. Use the Multi-ICE DLL configuration dialog from each debugger to select the different processors.

Note

If you are using an ARM debugger you can use the debugger session feature to avoid having to configure every debugger separately each time you start. Refer to *Configuring and debugging multiple processors* on page 4-27 for more information.

Using Multi-ICE with multiple processors is described in the following sections:

- *Controlling device execution*
- *Run control and the debugger* on page 3-26
- *About run control* on page 3-27
- *Setting up interaction between devices* on page 3-28
- *Setting up the poll frequency* on page 3-30.

3.4.1 Controlling device execution

In multiprocessor systems, the processors interact to produce the required results. The Multi-ICE server provides a feature called run control to help you debug these systems. Run control enables you to start and stop several processors at the same time in a configurable way. Using run control you can:

- Set several processors to a particular program location and state, and start them together. This is called *synchronous starting*. Multi-ICE ensures the processors start at exactly the same time, to within one **TCK** period.
- Force all the processors to stop if any of them hits a breakpoint, so you can see the collective state of the system when the breakpoint is reached. This is called *synchronous stopping*. The processors stop at almost the same time (limited by their response time to the stop request).

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You can also set up more complex start and stop conditions.

3.4.2 Run control and the debugger

Run control is a feature of the Multi-ICE server. Because the Multi-ICE server is the only place that has information about each of the processors in the system, it is on the server that you configure and control the interactions between different processors.

Note

- If you require run control to debug a specific part of your program, it is recommended that you initially set the server to **Independent** run control, so that no unexpected stop events occur. When the program reaches the area of interest, set up the desired run control settings in the server and start debugging.
- Make sure all the processors in the system are stopped when you change run control settings, because the settings are only applied when execution is started.

Synchronous starting

If you set up a synchronous start group that includes a particular processor, when you start that processor it does not start immediately. The debugger displays a message indicating that the server is waiting for other processors to start. You must use the other debuggers to start every processor in the synchronous start group before all of the waiting messages disappear and the processors all start together.

Note

If you require synchronous stopping as well as starting, you have to set up the server to do this before you start any of the processors in the group.

Synchronous stopping

When one of the processors stops (for example, due to a breakpoint or watchpoint), the debugger connected to that processor displays the processor state. If you have set up the Multi-ICE server to stop other processors together with this processor, the Multi-ICE server stops them and the debuggers connected to the other processors display the message:

```
Server stopped the processor
```

You can now examine the state of any processor in the group.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4.3 About run control

A processor can stop for a variety of reasons. These include:

- a debugger in `halt processor` mode connects to the processor
- the stop button is pressed in the debugger
- the processor hits a breakpoint or watchpoint
- the processor passes through a vector and vector catch is enabled
- the program running on the processor makes a semihosting call and standard semihosting is selected in the debugger
- the debugger steps to the next instruction
- external hardware, for example the ETM, asserts the **DBGRQ** signal on the core.

All of these stop events are detected by the server, and are dealt with according to your run control settings. When you use a setting other than **Independent** run control this means other processors might be stopped as well. This has a number of implications:

- If multiple device execution control is required while making semihosting calls, **DCC Semihosting** must be set in the Processor Properties dialog in AXD. In ADW, the debugger internal variable `semihosting_enabled` must be set to 2. See the description of `semihosting_enabled=2` in *Debugger internal variables* on page 4-36 for more details.

- The debugger uses breakpoints for its own purposes, but these are treated identically to your breakpoints by the Multi-ICE server. For example, if you create a breakpoint with a count value of 50 in AXD, the processor stops every time it hits the breakpoint, and then restarts until it has hit the breakpoint 50 times. This has the following consequences:
  - — Every time the processor stops, the server actions the stop events you have specified. This might not be the behavior you are expecting.
  - — Every time the processor is restarted (49 times in this example), the run control settings are applied. If you have set up synchronous starting and stopping, all the synchronously stopped processors stop each time the processor you are debugging hits the counted breakpoint. When the debugger tries to restart the breakpointed processor, the Multi-ICE server waits until the synchronously stopped processors are also started.

- When you click **Step** in the debugger, stop events are actioned for that processor. This is because the step is implemented by setting a breakpoint on the next instruction and then starting the processor. When the processor hits the breakpoint the server carries out stop events as in every other case.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Usually when you load an image using a debugger, a breakpoint is automatically placed on the function `main`. Clicking **Go** runs through the initialization code to this point and you must press **Go** again to run the program. This breakpoint is treated just like any other breakpoint.

3.4.4 Setting up interaction between devices

The run control dialog device interaction tab is shown in Figure 3-16. There can be several tabs of device TAP controller numbers that list all the available devices in blocks of four. The tabs enable you to set up the interaction between single devices or a range of devices.

On tabs containing fewer than four devices, `NOT VALID` is displayed for the unused boxes and the controls are grayed out.

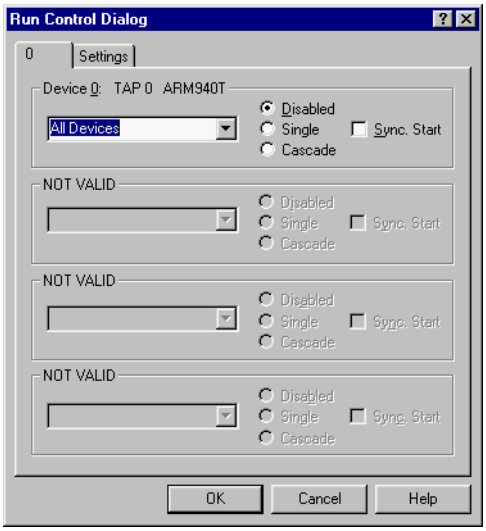


Figure 3-16 Setting up interaction between devices

Each device listed on the tab has a number of control settings:

Range field This is the drop-down list box directly beneath each of the device descriptions. It enables you to select any devices that are to be stopped by the current device. The default setting is **All Devices**.

If there are more than two devices available, you can select all devices, individual device numbers, or a range of devices.

For example, if you had 10 devices listed, you can stop devices 2, 5, 7, 8, and 9 using the following notation:

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
2, 5, 7-9
```

Disabled Stop events from this device are disabled and have no control over other devices.

Single Forces any devices in the range field that are currently running to stop. If any of the devices in the range field are forced to stop in this manner then no stop events for these devices are actioned.

Cascade Forces any devices in the range field that are currently running to stop. If any of the devices in the range field are forced to stop in this manner then any stop events for these devices are also actioned. If a device is not running and a stop event is actioned for that device, then the stop event is completely ignored.

Note

Stop events do not cascade through processors that are already stopped.

For example, assuming that initially devices one to ten are all running, Figure 3-17 shows that when device 1 stops, three other devices stop. This in turn stops a further set of devices.

Sync. Start When enabled, this processor is part of the synchronous start group. When one processor is started in the synchronous start group, the processor is marked ready to start. Every processor in the start group starts at the same time when they are all ready to start.

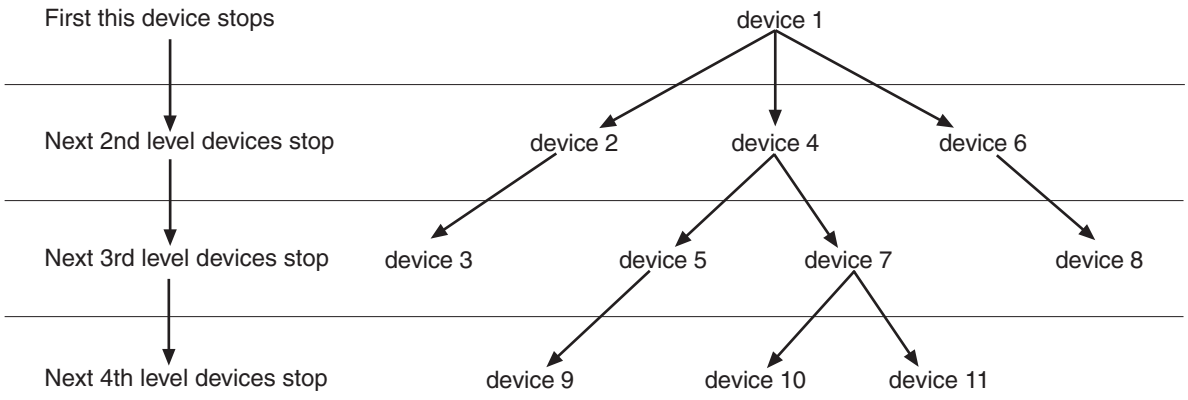


Figure 3-17 Cascade operation

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Combining settings

To provide finer control, you can combine settings to achieve individual results. Using the diagram in Figure 3-17 on page 3-29 as an example:

- If device 4 is disabled, devices 5, 7, 9, 10, and 11 are not affected by the control settings from device 1.
- If devices 4 and 5 are set to **Cascade**, but device 7 is disabled, the control settings from device 1 reach device 9, but not devices 10 or 11.
- If device 6 is set to **Single** instead of **Cascade**, device 8 is not affected by the control settings from device 1.

3.4.5 Setting up the poll frequency

The **Settings** tab on the **Run Control** dialog is shown in Figure 3-18. It displays a control that allows you to change the poll frequency that Multi-ICE uses. You can do this to find a balance between the responsiveness of your host computer and the number of times Multi-ICE polls the devices to find out their status.

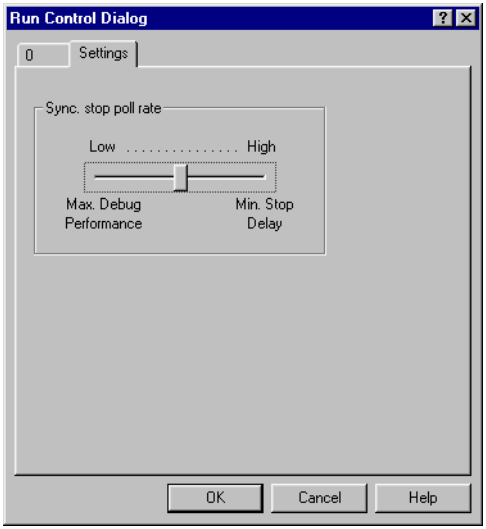


Figure 3-18 Setting up the poll frequency

There is a delay between a processor entering debug state and Multi-ICE noticing this and acting upon it. The settings are:

Low Good debugger responsiveness, with minimal status polling. If the processor stops it might not be noticed quickly.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

High Fast status polling, with very poor debugger responsiveness. If the processor stops it is noticed quickly.

The default setting is midway on the scale.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 4
Debugging with Multi-ICE

This chapter describes how to connect the Multi-ICE DLL to the supported ARM debuggers, and describes the features of the Multi-ICE DLL.

This chapter must be read in conjunction with the manuals for the debugger you are using, for example the *ADS Debuggers Guide* and the *ADS Debug Target Guide*. It contains the following sections:

- *Compatibility with ARM debuggers* on page 4-2
- *Connecting Multi-ICE to ADW, ADU, or AXD* on page 4-3
- *Configuring the Multi-ICE DLL* on page 4-8
- *Configuring and debugging multiple processors* on page 4-27
- *Debugger internal variables* on page 4-36
- *Post-mortem debugging* on page 4-45
- *Access to CP15* on page 4-49
- *Semihosting* on page 4-50
- *Watchpoints and breakpoints* on page 4-55
- *Cached data* on page 4-60
- *Debugging applications in ROM* on page 4-62
- *Accessing the EmbeddedICE logic directly* on page 4-65.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.1 Compatibility with ARM debuggers

To debug your ARM-targeted image using any of the ARM debugging systems, you can use any of the following ARM debuggers:

- *ARM eXtended Debugger* (AXD) for Windows or UNIX
- *ARM Debugger for Windows* (ADW) (SDT 2.51, ADS v1.0.1, or ADS v1.1)
- *ARM Debugger for UNIX* (ADU) (SDT 2.51, ADS v1.0.1, or ADS v1.1).

The debugger works in conjunction with Multi-ICE. Multi-ICE provides the ability to access the target, and tools to configure the debugger to access the target in the correct way. The debugger provides the user interface items such as register windows and disassemblers that make it possible to debug your application.

Refer to the documentation supplied with your target board for more information on development boards.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.2 Connecting Multi-ICE to ADW, ADU, or AXD

After loading a configuration file or using **Auto-Configure**, the debugger must be connected to the Multi-ICE server. The procedure for the ARM debuggers varies and is described in the following sections:

- *Connecting AXD* on page 4-4
- *Connecting ADW and ADU* on page 4-5.

The procedure is identical for Windows and UNIX based systems, so although the screen shots presented are from Windows, the same instructions apply to ADU and AXD running on Solaris, HP-UX, and Linux.

Note

This note only applies to workstations running a version of Microsoft Windows.

By default Windows Explorer, and therefore the file open dialog box, hides files with the file extension `.dll`. As a result, unless this setting is changed, the Multi-ICE DLL is not shown.

To show `.dll` files in Windows 95 or Windows NT 4.0 without the Desktop Update:

1. Open a Windows Explorer Window and select **View** → **Options**.
2. Select the **View** tab.
3. Select the **Show all files** radio button.
4. Click on the **Options** dialog **OK** button.

To show `.dll` files in Windows 95 or Windows NT 4.0 with the Desktop Update, and Windows 98, Windows ME, or Windows 2000:

1. Open a Windows Explorer Window and select **View** → **Folder Options**.
2. Select the **View** tab.
3. Within the tree view, find **Files and Folders** → **Hidden Files**. Select the **Show all files** radio button in that group.
4. Click on the **Folder Options** dialog **OK** button.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.2.1 Connecting AXD

Use this procedure to activate the Multi-ICE DLL within AXD using Windows or UNIX.

1. Select **Options → Configure Target** as shown in Figure 4-1.

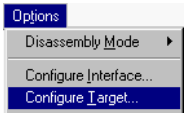


Figure 4-1 The AXD Options menu

This displays the **Choose Target** dialog shown in Figure 4-2.

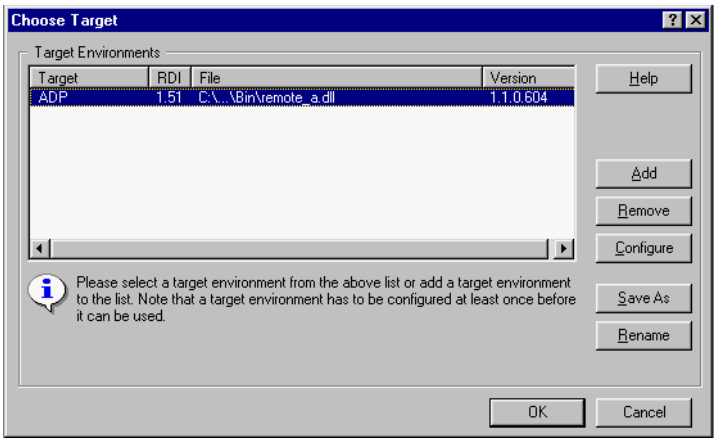


Figure 4-2 The AXD Choose Target dialog

2. If Multi-ICE is listed in the **Target Environments** list, select it (by clicking on it) and go on to step 3. If it is not listed:
   a. Select **Add**. A Windows **Open** dialog is displayed.
   b. Navigate to the Multi-ICE install directory (for example, `C:\Program Files\ARM\Multi-ICE`).
   c. Find the file `Multi-ICE.dll`, select it, and click on the dialog **Open** button as shown in Figure 4-3 on page 4-5.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

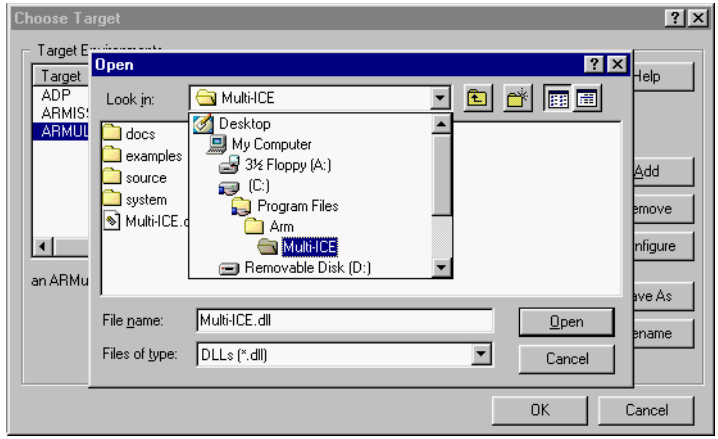


Figure 4-3 Selecting the Multi-ICE DLL using AXD

Clicking on **Open** dismisses the dialog box.

The file path name of the Multi-ICE DLL is displayed in the **Target Environments** list.

3. Select **Configure** to display the Multi-ICE configuration dialog.
4. Now go to *Configuring the Multi-ICE DLL* on page 4-8.

4.2.2 Connecting ADW and ADU

Use this procedure to activate the Multi-ICE DLL within ADW or ADU.

1. Start ADW or ADU. The last debug target used, or ARMulator® if the debugger has just been installed, is activated automatically. If the Target Warning dialog appears, click on **No**. This causes ADW or ADU to activate ARMulator.
2. Select **Options** → **Configure Debugger** as shown in Figure 4-4.

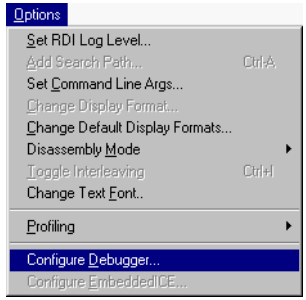


Figure 4-4 The ADW and ADU Options menu

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The debugger displays a Debugger Configuration dialog similar to Figure 4-5. Select the **Target** tab.

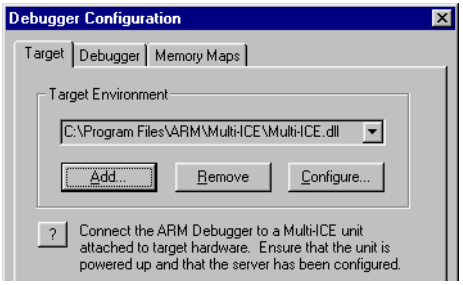


Figure 4-5 ADW configuration dialog with Multi-ICE active

3. If Multi-ICE is listed in the **Target Environment** list as shown in Figure 4-5, go on to step 4.

   If Multi-ICE is not listed:

   a. Select **Add**. The Open dialog is displayed.

   b. Navigate to the Multi-ICE install directory (for example, `C:\Program Files\ARM\Multi-ICE`).

   c. Find the file `Multi-ICE.dll`, and select it as shown in Figure 4-6. Click on the dialog **Open** button.

      Clicking on **Open** dismisses the **Open** dialog and the file path name of the Multi-ICE DLL is displayed in the **Target Environment** list.

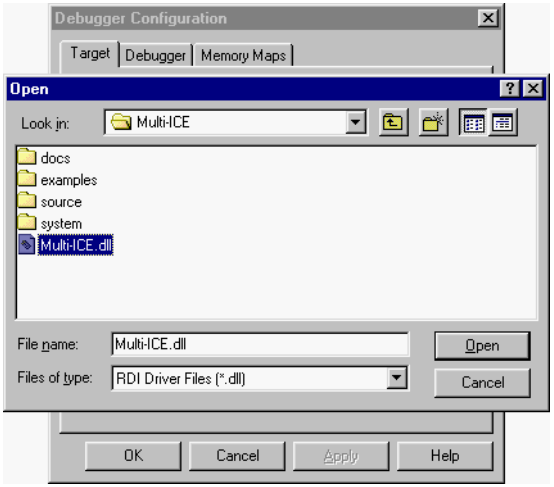


Figure 4-6 Selecting the Multi-ICE DLL using ADW

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. Select **Configure** in the Debugger Configuration dialog. This displays the Multi-ICE Configuration dialog.
5. Now go to *Configuring the Multi-ICE DLL* on page 4-8.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3 Configuring the Multi-ICE DLL

This section describes the elements of the Multi-ICE configuration dialog. The dialog box is tabbed, using the following headings:

- *Connect configuration tab*
- *Processor Settings Tab* on page 4-14
- *Advanced configuration tab* on page 4-18
- *Board configuration tab* on page 4-21
- *Trace configuration tab* on page 4-22
- *About Multi-ICE tab* on page 4-23
- *Channel viewer configuration tab* on page 4-24.

The last section describes how the configuration settings are stored:

- *Persistence of DLL settings* on page 4-26.

4.3.1 Connect configuration tab

The Multi-ICE server connection configuration dialog is shown in Figure 4-7. You use it to select the Multi-ICE server and processor to debug.

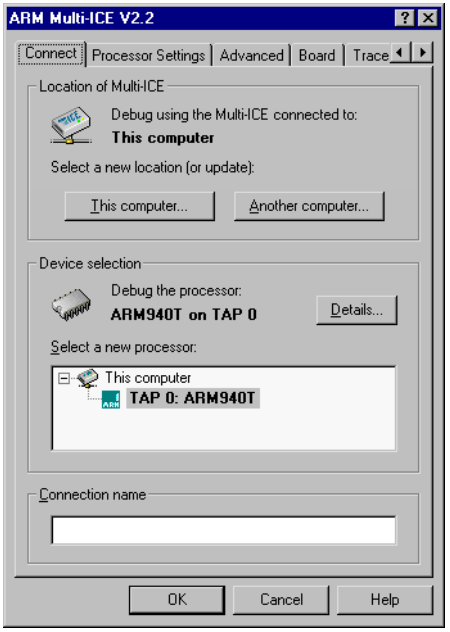


Figure 4-7 Multi-ICE Configuration dialog

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If you have not yet successfully configured Multi-ICE, the Welcome to Multi-ICE dialog, shown in Figure 4-8, is also displayed. After reading it, dismiss it by clicking **OK**.

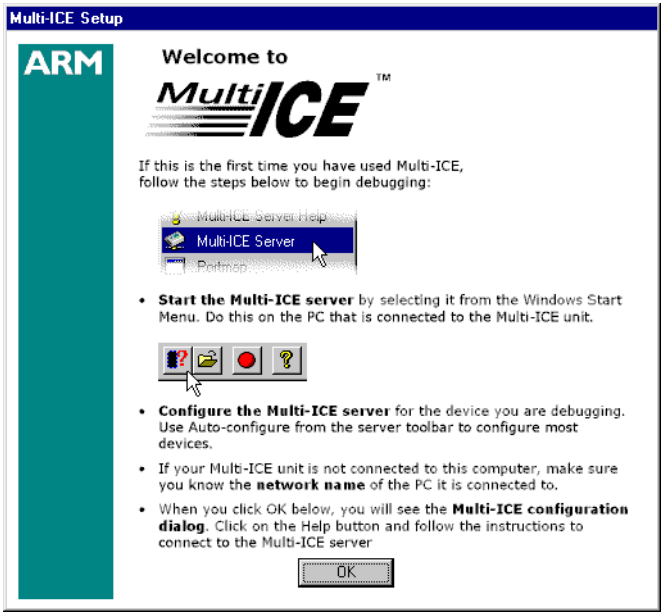


Figure 4-8 Multi-ICE Welcome dialog

The configuration dialog includes the following items:

Location of Multi-ICE

You must enter the name of the workstation that Multi-ICE contacts to find the Multi-ICE server and the Multi-ICE interface unit. If you:

- have an existing connection, the name of the workstation is shown here, and the device details are shown in Device selection
- have not connected to a server, this area and the Device selection area are empty.

Click on the button labeled **This computer...** if the Multi-ICE interface unit is connected to the workstation you are using. If there is no Multi-ICE server running, the software asks if it should start one.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Click on **Another computer...** if the server is running on a different workstation. The **Select server location** dialog appears, and you can enter the name of a server in **Network address** or select it from the network list. See *Remote Multi-ICE servers* on page 4-11 for more information.

Device selection

You must select the desired processor device (or core) from the device tree. The devices correspond to those shown in the TAP configuration area on the Multi-ICE server window, with device aliases shown as subordinate to the main device.

If you require more information on the selected device, click **Details...** to display the Driver Details dialog (Figure 4-9).

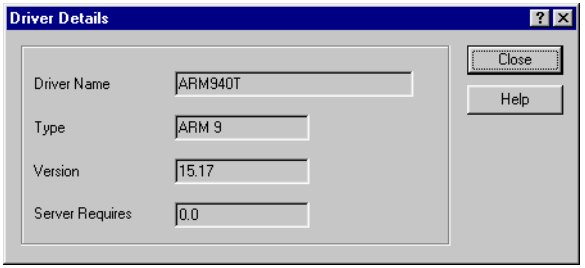


Figure 4-9 Driver Details dialog

The fields contain the following information:

Driver Name The name Multi-ICE uses to refer to the device. This is the same name that is used in the driver information file `IRlength.arm`.

Type The processor type, for example, ARM 7, ARM 9, XScale.

Version The version number of the software driver used to control the device.

Server Requires The version of the Multi-ICE server required to use this driver.

Connection name

An entry here is optional. If you want to you can enter a name for this connection. This helps you identify the engineers or test programs that are using the device. The name is displayed in the debug pane on the server when the connection is made.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To accept your settings and connect to the target processor:

1. Click on **OK**. You return to the Debugger Configuration dialog.
2. Click **OK** in the configuration dialog to connect to the target processor. If the connection is successful, the device name turns red in the server window and connection information is displayed in the server console window. Otherwise, an error message is displayed.

Remote Multi-ICE servers

The **Another computer...** button on the Connect configuration tab (see *Multi-ICE Configuration dialog* on page 4-8) configures Multi-ICE to connect to a Multi-ICE server on another computer. The dialog box that is displayed depends on the network software that is configured on your workstation:

- if you have a Unix workstation, or a Windows workstation but no local area network software installed, refer to *Workstations with no Network Neighborhood*
- if you have a Windows workstation and network software and can use the Windows Network Neighborhood, refer to *Workstations with a Network Neighborhood*.

Workstations with no Network Neighborhood

On UNIX workstations, and Windows workstations that do not have the Windows Computer Browser service available, a dialog box is displayed so that you can type in the name of a machine to connect to.

If you are using a TCP/IP network with Windows 95, Windows 98, or Windows Me, you might have to install the Windows Computer Browser service before the network browse dialog shown in Figure 4-10 on page 4-12 can be used.

Workstations with a Network Neighborhood

On Windows workstations that have the Windows Computer Browser service available, the dialog box shown in Figure 4-10 on page 4-12 is displayed, enabling you to browse the Network for Multi-ICE servers to connect to. Only workstations that have both the Windows Computer Browser service and a remotely accessible Multi-ICE server running are shown in the browser.

You can use the dialog box in two ways:

- If you know the workstation you want to connect to, you can enter its name in the **Server name** text field. You can enter either the textual name (for example, `PC2`) or the IP address (in dotted quad form, for example `192.168.3.1`). Click on **OK** to finish.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- You can search for servers using the tree view.

To browse your network, expand the names in the list area by clicking on the ⊞ icon until you see the names of workstations. These have a Multi-ICE server, with the **Allow Network Connections** setting enabled, running on them.

To list the Multi-ICE servers in a group, every workstation in the group has to be contacted. Multi-ICE contacts many workstations at a time, displaying a message at the top of the dialog box as it does so.

If no workstations in the group have a server running on them that can be contacted, the group name is displayed without the ⊞ icon.

When a machine is displayed, selecting the ⊞ icon next to its name displays the list of processors currently configured on that server.

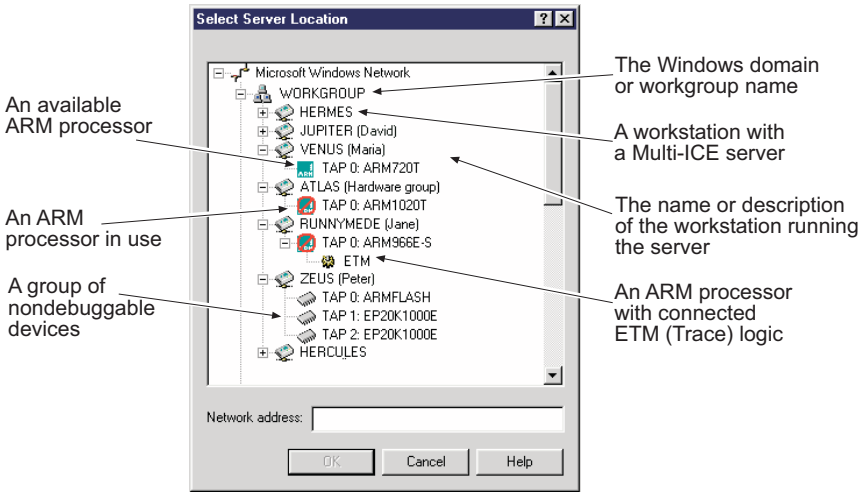


Figure 4-10 Server Browse dialog

The icon beside the device name indicates the current connection state:

- A red circle over the icon indicates a device that has an active connection.
- A turquoise `ARM powered` icon indicates an ARM device that Multi-ICE can debug.
- A indicates a device that Multi-ICE cannot debug, for example an FPGA, a DSP core or a Flash memory device.
- A indicates an extension device connected to the same TAP controller, for example an *Embedded Trace Macrocell* (ETM). See the *Trace Debug Tools User Guide* for more information for more information about the ETM.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If you select a device to connect to and then click on **OK** before the list is complete, no further information is added to the display. However, current activity must be allowed to complete, and while this is happening, the message text starts with `Stopping...`. It might take several seconds for this process to complete.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.2 Processor Settings Tab

The Processor Settings tab enables you to change processor specific settings before you connect to the target. The settings you can define depend on the processor you are connecting to, and so you must select a processor (using the Connect tab) before this tab can be used. You can change the following settings:

Cache clean code address

This setting, shown in Figure 4-11 on page 4-15, is the base address of an area of 128 bytes of memory that is used to store a code sequence that ensures that all of the *dirty data* in the Data Cache (DCache) is written into main memory.

Cleaning the DCache is important because when the processor caches are enabled, program instructions that are written to target memory are written through the DCache but read through the instruction cache. If the DCache is not cleaned, some or all of the instructions are not written to main memory before the processor executes them, and so the previous contents of memory at that address is executed instead.

Multi-ICE loads the cache clean code as required. The area must be program memory, readable and writable, and must not be used for any other purpose. If Multi-ICE cannot load code to this memory you see the error `Could not clean D-Cache - memory may appear incoherent in writeback regions`. Refer to *Multi-ICE server messages* on page 5-12 for more information about this message.

For some processor types, restart code must also be downloaded that enables Multi-ICE to restart the processor when a breakpoint or exception is handled. For these processors, the restart code is written to the cache clean code address.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Figure 4-11 Multi-ICE Processor Settings tab showing cache setting

Cache clean data address

This setting for XScale microarchitecture processors, shown in Figure 4-12 on page 4-16, is the base address of a 32KB region of memory that Multi-ICE uses when cleaning the processor cache. This memory region:

- must be cachable
- must be aligned to a 32KB address (that is, the least significant 15 bits of the address must be zero)
- must be 32KB in size.

Note

Physical memory does not have to exist at this location.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

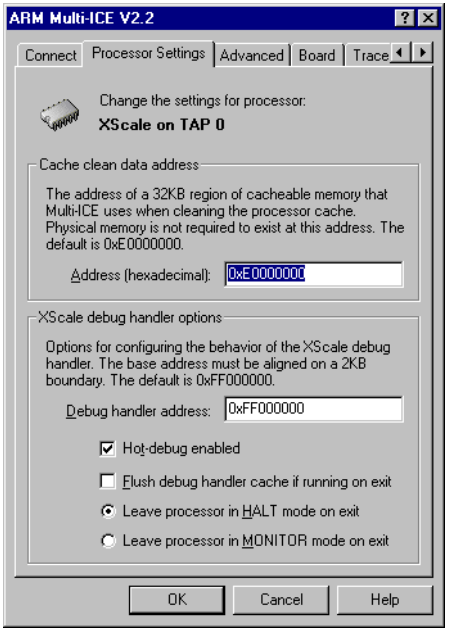


Figure 4-12 Multi-ICE Processor Settings tab showing XScale settings

XScale debug handler options

These settings, shown in Figure 4-12, configure the behavior of the debug handler that is used with an XScale microarchitecture processor.

The following settings are available:

Debug handler address

The base address of a region of memory that can be used by the debug handler code. This memory:

- must be aligned to a 2KB address (that is, the least significant 11 bits of the address must be zero)
- must be 2KB in size
- must be in the range of an ARM branch instruction (approximately ±32MB) from address `0`
- must not be used for any other purpose.

Note

Physical memory does not have to exist at this location.

If you enter an invalid address a message box reports that `The address you have entered is invalid`.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Hot-debug enabled

A toggle that enables or prevents Multi-ICE from connecting to an already running XScale microarchitecture processor by using a debug handler:

- If you enable hot-debug, ARM Limited recommends that the target provides firmware support for this feature:
  - For details of the code that is required, see *Debug handler firmware support* on page D-7.
  - If you do add this firmware support, you must check the **Flush debug handler cache if running on exit** box, and select the **Leave processor in Monitor mode on exit** button.
- If you disable hot-debug, Multi-ICE always resets the processor when connecting.

Flush debug handler cache if running on exit

A toggle that controls whether or not the debug handler is flushed from the cache on exit from the debugger:

- If this box is checked, and the processor is running, the debug handler is flushed from the cache. The debugger cannot later reconnect to the handler.
- If this box is not checked, the debug handler is not flushed from the cache. The debugger can later reconnect to the handler.

Note

Any code that subsequently alters the exception vectors does not function correctly (see *Intel XScale microarchitecture processors* on page D-3).

Leave processor in Halt mode on exit, Leave processor in Monitor mode on exit

Buttons that set whether to leave the processor in Halt or Monitor mode on exit from the debugger:

- If you leave the processor in Halt mode, and a reset subsequently occurs, the debug handler is not flushed from the cache. The processor halts, waiting for the debugger to connect to the handler.
- If you leave the processor in Monitor mode, and a reset subsequently occurs, the debug handler is flushed from the cache, and the processor resets.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.3 Advanced configuration tab

The Advanced configuration tab contains items that enable you to configure the debugger to match the memory endianness of your target, whether to cache information, and the debugger software interface method. The tab is shown in Figure 4-13.

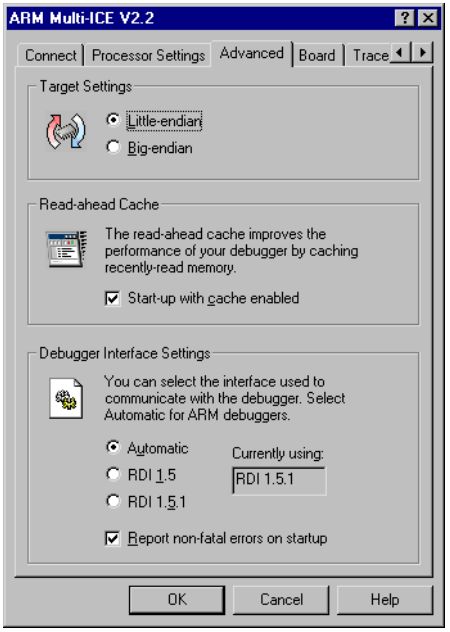


Figure 4-13 Multi-ICE Advanced settings tab

The dialog box contains the following items:

Target Settings

You can specify whether the target is **Little-endian** or **Big-endian** using the **Target Settings** radio buttons in the target configuration dialog.

These buttons are unavailable if you are using older software, such as SDT 2.51 or ADU. You must instead:

1. Choose **Configure Debugger** from the **Options** menu of your debugger.
2. Click the **Debugger** tab.
3. Select one of the **Endian** buttons.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Read-ahead Cache

This is a check box allowing you to enable or disable caching target memory in the host while the processor is stopped. The checkbox setting is the initial value of `internal_cache_enabled`. You can flush the cache at a specific time by setting the debugger internal variable `internal_cache_flush` to 1. You can switch the cache off temporarily by setting the debugger internal variable `internal_cache_enabled` to 0. See *Internal variable descriptions* on page 4-39 for more information.

Read-ahead caching improves memory read performance by reading more memory than the debugger requests, and caching the rest in case it is required. This improves performance considerably for some operations, such as when you are stepping through code with a lot of string variables displayed in a debugger window.

Initially the DLL does not read ahead. When the first successful read request is made for a region of memory, the DLL learns it can safely access that region. It then caches that region until the debugger restarts.

All of the ADS debuggers save the read ahead setting between sessions, but the version of ADW in SDT 2.51 does not.

Note

Read-ahead caching is on by default. You must switch this feature off if:

- you are trying to debug a system with demand paged memory
- you are using the debugger to read hardware registers, and you do not want to unintentionally read neighboring registers.

Debugger Interface Settings

Multi-ICE supports debuggers that conform to RDI 1.5.1 and also allows you to connect to debuggers that conform to RDI 1.5. This item shows you the RDI version that is currently in use.

You can select the RDI version to be used from the dialog box.

Automatic Selects the most appropriate mode for the debugger, using an automatic version negotiation scheme. This is the default setting.

RDI 1.5 Forces the Multi-ICE DLL to use RDI version 1.5.

RDI 1.5.1 Forces the Multi-ICE DLL to use RDI version 1.5.1.

If you are using an ARM debugger, it is recommended that you use the **Automatic** setting, because this works correctly with all ARM debuggers.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

AXD only supports RDI 1.5.1, and **Automatic** version negotiation selects RDI 1.5.1. If you force AXD to connect to Multi-ICE using RDI 1.5, the AXD **Stop** button does not work.

Report non-fatal errors on startup

This setting relates to errors that occur while the debugger is first connecting to a Multi-ICE target. If you check this option, then both fatal and non-fatal errors are reported. This is the default, and means that you are informed of any problems that Multi-ICE detects.

Some debuggers consider that any error detected during configuration is fatal, and this can prevent you from being able to connect to your target. If this happens, you must uncheck this item, so that Multi-ICE only reports fatal errors.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.4 Board configuration tab

The Board configuration tab has a list of target boards. When you select a board, the debugger then shows the memory mapped registers for the peripherals and microcontrollers on that board.

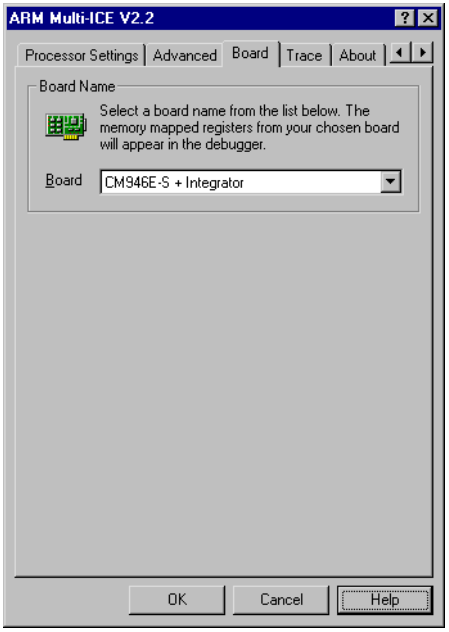


Figure 4-14 Board tab

Note

This tab only appears if you are using a recent toolset that supports this functionality (such as ADS 1.2 or later), but you are not using RealMonitor.

If you are using RealMonitor, you can instead set the board that you are using with RealMonitor.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.5 Trace configuration tab

The Trace tab is displayed, as shown in Figure 4-15, when the ARM *Trace Debug Tools* (TDT) have been installed on your workstation. If TDT is not installed, the Trace tab is not shown.

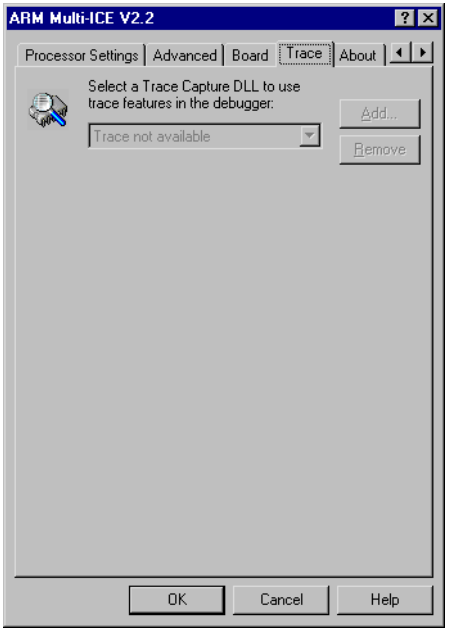


Figure 4-15 Trace configuration tab

The Trace tab enables you to configure the trace component that you use to capture trace information. For more information on the TDT, including how to use the Multi-ICE Trace configuration tab, refer to the *Trace Debug Tools User Guide*.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.6 About Multi-ICE tab

Information about the version of Multi-ICE that you are using is displayed on the **About** tab shown in Figure 4-16. It displays the full version information of the Multi-ICE DLL that you are using with the components that are connected to it.

If TDT is installed and configured, the version number of the DLL used to control trace capture, for example, `multitrace.dll`, is included in this list.

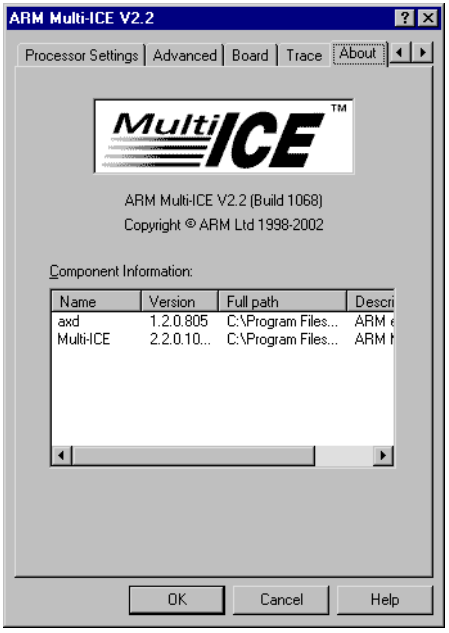


Figure 4-16 About tab

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.7 Channel viewer configuration tab

Channel viewers allow information transferred across the DCC to be manipulated by a program external to the debugger. There is a viewer supplied with ARM debuggers, called ThumbCV, that interprets the words sent over the DCC and displays them in a window. It also provides a text entry area that enables characters to be sent to the target. For more information refer to the *ADS Developer Guide*.

To see how Multi-ICE buffers the information passed over the DCC channel, refer to *Channel viewer buffering in Multi-ICE* on page 4-25.

Using channel viewers with AXD

The Multi-ICE configuration dialog **Channel Viewers** tab is only available when using ADW. The use of channel viewers in AXD is described in the AXD documentation (see the *ADS Debuggers Guide*).

Using Channel viewers with ADW

Channel viewers can enable the host to monitor the target in more complex ways, or simulate the presence of external sensors. See the *ADS Debuggers Guide* for more information on channel viewers. The DCC hardware is described in detail in the technical reference manual for the ARM core you are using.

The channel viewer controls shown in Figure 4-17 enable or disable the selected channel viewer DLL.

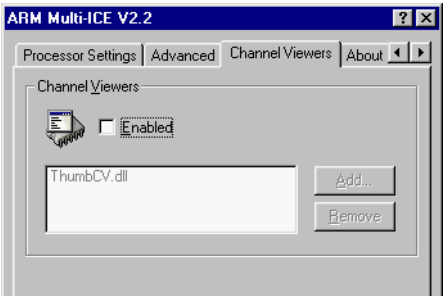


Figure 4-17 Channel viewer controls

Note

- You cannot use the channel viewer and DCC semihosting (by setting `semihosting_enabled = 2`) at the same time because they use the same DCC. Therefore, if you have DCC semihosting enabled, attempting to add or use a channel viewer fails.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- You cannot use DCC channel viewers with the ARM10 (Rev 0) processor or with XScale microarchitecture processors.

There are three controls. The **Enabled** checkbox enables the channel viewer functions (and also disables other uses of the DCC). When **Enabled** is checked, the two buttons **Add** and **Remove** allow you to manipulate the list of channel viewer DLLs that are available to Multi-ICE:

Add Adds a channel viewer DLL.

Remove Removes the selected DLL.

Adding the viewer causes the channel viewer to be initialized and, in the case of the supplied ThumbCV viewer, this creates a new window on the screen.

Channel viewer buffering in Multi-ICE

The Multi-ICE DLL has an internal 1024-word buffer for DCC transfers from the debugger to the target. Data being sent to the target is cached in this buffer until the target program requests it.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3.8 Persistence of DLL settings

The persistence of the DLL settings between sessions, and how they are stored, depends on the debugger you are using:

ADW (SDT 2.51) Saves most of the DLL session settings in the registry. However, it does not save session settings that have been introduced since Multi-ICE Release 1.3, such as the state of the read-ahead cache enabled check box.

ADW (ADS 1.0.1), ADU (ADS 1.0.1)

Saves all the DLL session settings in a file within the user profiles. For example, on Windows NT the filename is:

`c:\winnt\profiles\`*`username`*`\adwtoolconf-`*`default`*`.cnf`

If you use the `-session` *`name`* parameter to start ADS 1.0 ADW, the settings for each session are stored in different files, where `default` is replaced by *`name`*.

AXD (ADS 1.0.1) Saves all the DLL session settings in the registry.

AXD (ADS 1.1 or later)

Saves all the DLL session settings in the registry, or in the file you specify when you select **File → Save Session**.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.4 Configuring and debugging multiple processors

This section describes setting up a multiple processor system using the AXD debugger from ADS v1.1 (or a later version), and Multi-ICE Version 2.1 (or later). It is split into the following sections:

- *Configuration using named AXD target configurations*
- *Configuration using session files* on page 4-29.

Select the method that best suits the way you work. Multiple targets is the simpler method, but session files enable a more automated setup.

Note

To use AXD with a multiple processor system connected to the Multi-ICE interface unit, you must run an instance of AXD for each processor that you want to debug. It is not possible to connect one instance of AXD to more than one processor.

4.4.1 Configuration using named AXD target configurations

AXD enables you to use one or more named target configurations. This is useful in a multiple processor setup, because you can create one target for each processor and switch between them easily.

To set this up:

1. Run AXD.
2. Select **Options → Configure Target...** to display the target configuration dialog.
3. Add Multi-ICE to the list of targets using the **Add** button if it is not already present, as described in *Configuring the Multi-ICE DLL* on page 4-8.
4. Copy the Multi-ICE target configuration to create one target configuration for each processor on the target board, as follows:
   a. Select the Multi-ICE target in the list.
   b. Click **Save As** to display the target configuration Save dialog.
   c. Enter the new name of the target configuration, as shown in Figure 4-18 on page 4-28.
   d. Click **OK**.

   For example, in a three processor setup you can create two more copies of Multi-ICE called `Multi-ICE_TAP1` and `Multi-ICE_TAP2`, and then you can click **Rename** to rename the original to `Multi-ICE_TAP0`.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

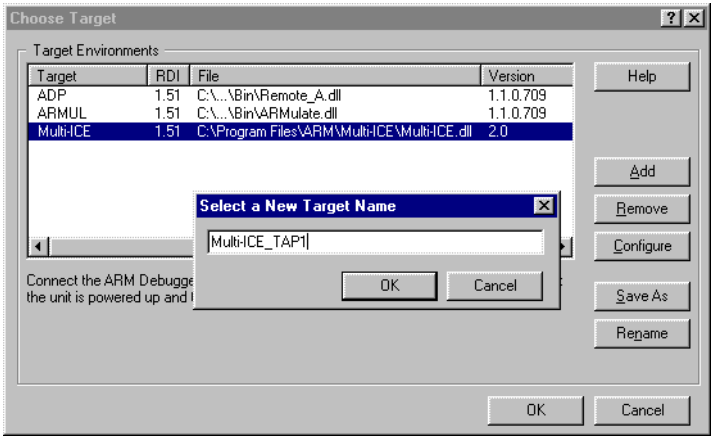


Figure 4-18 Saving a named target configuration

5. Configure these targets separately by selecting the name in the Target Environments list and clicking **Configure**. Refer to *Configuring the Multi-ICE DLL* on page 4-8 for more information.

6. Click on OK in the AXD Configure Target dialog when you have completed the configuration of each target. AXD will save all the settings for all the targets and connect to the one you selected.

You can now easily swap between the processors in your system by selecting a different target in the target configuration dialog.

Configuring AXD to select a target on startup

In its default configuration, when you run AXD it automatically selects the target that was last in use when you run it. You can change this behavior, so that it brings up the target configuration dialog on startup, rather than connecting to the default target. To do this:

1. Ensure the Multi-ICE server is correctly configured for your target board.

2. Run AXD.

3. Select **Options → Configure Interface…**, to display the Interface Configuration dialog.

4. Disable the **Reselect Target** option on the **Session File** tab.

5. Exit AXD.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Starting AXD from CodeWarrior

The CodeWarrior IDE supplied with ADS provides a button that builds your project and then runs it by starting up AXD automatically. If you have multiple processors AXD always tries to run your project on the most recently used processor. It is therefore recommended that you make the changes described in *Configuring AXD to select a target on startup* on page 4-28.

4.4.2 Configuration using session files

If you require a more automated way of running several debuggers, you can use the session feature in AXD. An AXD session file includes debugger settings, for example the name of an executable image file and the locations of windows, and also the current target configuration. A command-line argument to AXD enables you to select a particular session file on startup. In this way you can automatically connect AXD to a particular processor by changing the session file you use.

This method is less flexible than the multiple target procedure described in *Configuration using named AXD target configurations* on page 4-27, especially if you frequently change the processor you connect to. However, specifying a processor when you start AXD enables a greater level of automation.

To configure the first AXD session file:

1. Ensure the Multi-ICE server is correctly configured for your target board.
2. Run AXD.
3. Select Multi-ICE as the target configuration. It is recommended that you click **Remove** to delete all but one copy of the Multi-ICE configuration, to avoid confusion in making any further changes.
4. Configure Multi-ICE to connect to one of the target board processors.
5. Select **Options → Configure Interface...**.
6. Enable the **Reselect target** option in the **Session File** tab.

   If you do not have this option enabled, the session files you create will not select any target when you load them back into AXD.
7. Click **OK**.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Changing session settings

You must now create a session file for each processor in your system. For each processor on your target board:

1. Select **Options → Configure Target...**
2. Select Multi-ICE in the target configuration list
3. Configure Multi-ICE for that processor.
4. Click OK in the Multi-ICE configuration dialog to accept the configuration.
5. Click OK in the Configure Target dialog to connect to the processor.
6. If you want AXD to load a specific executable image when it connects to this processor:
   a. Click **File → Load Image** to load the file.
   b. Select **Options → Configure Interface...**.
   c. Enable the **Reload Images** option in the **Session File** tab (Figure 4-19 on page 4-31).
   d. Click **OK**.
7. If you want to run a configuration script as AXD starts up, for example to set the processor into a particular state:
   a. Select **Options → Configure Interface...**.
   b. Select **Run Configuration Script** in the **Session File** tab.
   c. Use the **Browse...** button to locate the configuration script, or type the name into the text field, as shown in Figure 4-19 on page 4-31.
   d. Click **OK**.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

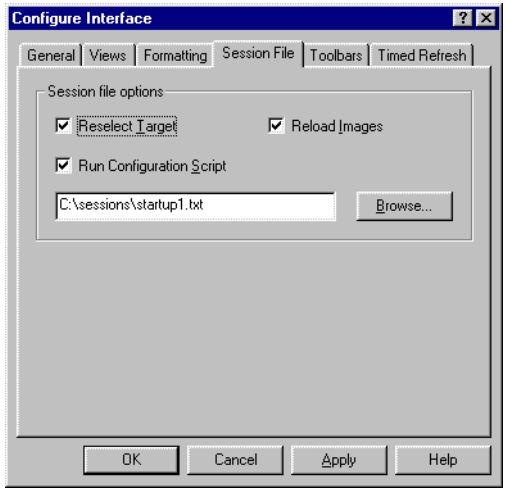


Figure 4-19 Configuring AXD to run a configuration script

8. Save the session file by selecting **File → Save Session…**. You must use the file extension `.ses`. It is recommended that you use a name that refers to the processor this configuration connects to.

   It is recommended you save session files together in an easily-accessible directory.

You can now select which processor to connect to by specifying the appropriate session file in AXD's command line. For example:

```
axd -session C:\sessions\tap1.ses
```

An example of three AXDs connected to a Multi-ICE server and a multiple processor target is shown in Figure 4-20 on page 4-32.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

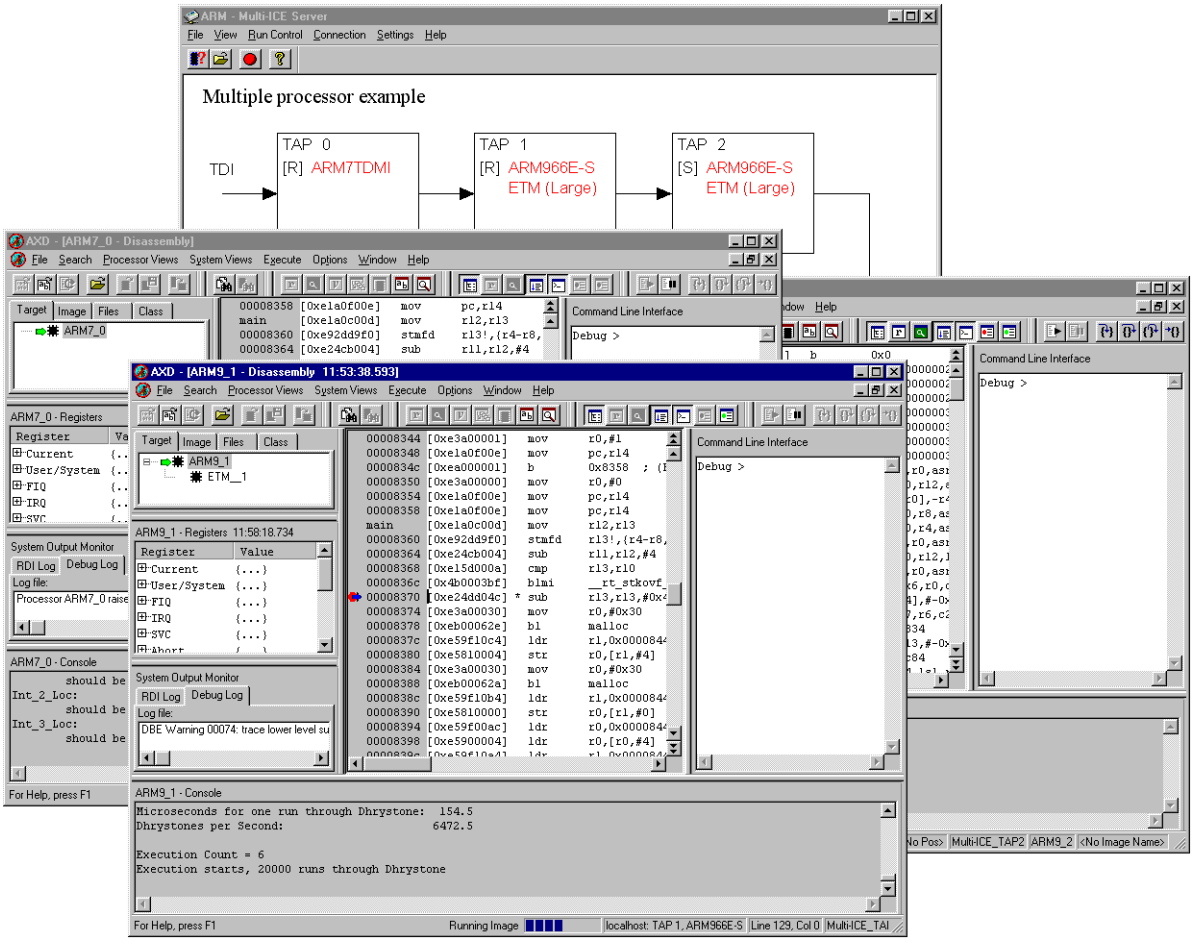


Figure 4-20 Three AXDs and the Multi-ICE server configured for a multiple processor target

By putting several AXD session file commands into a script, you can automatically start up the required number of instances of AXD, one connected to each processor. Example files are shown in Example 4-1 on page 4-33 and Example 4-2 on page 4-33. The Unix script file command `source ads.cshrc` sets up the path and environment for the ADS executables. The Windows versions of these files are in the examples directory of your Multi-ICE installation.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

Under Windows the `pause` command is used. This is because several copies of AXD cannot be loaded concurrently under Windows. The `pause` command waits for you to press a key, and you must do this after each copy of AXD has loaded and initialized.

Example 4-1 Windows batch file

```
start axd -session C:\sessions\tap0.ses
pause
start axd -session C:\sessions\tap1.ses
pause
start axd -session C:\sessions\tap2.ses
```

Example 4-2 Unix shell script

```
source ads.cshrc
axd -session sessions/tap0.ses &
axd -session sessions/tap1.ses &
axd -session sessions/tap2.ses &
```

You can also create a shortcut on your desktop for each processor. To do this under Windows, you must:

1. Right click on the desktop to bring up the context menu
2. Select **New → Shortcut**
3. In the Create Shortcut dialog, click **Browse**
4. Use the browse dialog to locate AXD.exe. If you installed AXD in the default location, it is in `C:\Program Files\ARM\ADSv1_1\Bin\axd.exe`.
5. Click **Open**.
6. Press the **End** key to move the cursor to the end of the line, and type a space and then:

   `-session “C:\sessions\tap0.ses”`

   The line should now look something like:

   `"C:\Program Files\ARM\ADSv1_1\Bin\axd.exe" -session “C:\sessions\tap0.ses”`
7. Click **Next>**

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8. Enter a name for the shortcut, for example, `AXD on Tap0`.
9. Click **Finish**.

Double clicking on this shortcut launches AXD and automatically connects to the configured processor with the settings you saved in the session file.

You can configure your Unix desktop in a similar way. Refer to the system documentation for your desktop environment (for example, CDE or OpenWindows) for more information.

Note

If you want to change any of the settings in a session file, you must re-save it. AXD does not automatically update the session file it was started from if you change settings.

Using sessions from CodeWarrior

When you have set up your sessions, you might also want to configure CodeWarrior so that clicking on the Run button automatically selects one of your sessions. This works best if you have a different CodeWarrior project for each processor. Otherwise you have to change the project settings each time you want to use a different processor.

To configure a project to load into AXD with a particular session:

1. In the CodeWarrior IDE, display the project settings dialog and in the settings pane tree control select ARM Runner in the Debugger section.
2. Select the **Choose Debugger** tab.
3. Select **AXD**.
4. In the **Equivalent Command Line** control, enter the `-session` parameter as described in *Changing session settings* on page 4-30.

   For example:

   ```
   axd -session C:\sessions\tap0.ses -exec &1
   ```

   Note

   The `-session` parameter and the session file name must be together and must precede a `-debug` or `-exec` parameter on the command line.
5. Click on **Save** and close the dialog.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Caution

If you subsequently change any of the other settings in this panel you must enter the `-session` argument again.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.5 Debugger internal variables

The debugger internal variables that are always present are described in the relevant debugger user guides. This section describes additional debugger internal variables that become available when you install the Multi-ICE software. This section includes information on:

- *Accessing debugger internal variables*
- *Internal variable support by processor*
- *Internal variable descriptions* on page 4-39.

4.5.1 Accessing debugger internal variables

Debugger internal variables are values that control the behavior of the debugger or the way it accesses the target. Some variables control the front-end debugger (for example, `searchpath`). Others enable you to control the operation of Multi-ICE without using the Multi-ICE configuration dialog.

In AXD, the variables `vector_catch`, `vector_address`, `semihosting_enabled`, and `semihosting_dcchandler_address` are all accessed using the **Properties** menu item on the right mouse button menu from the processor icon.

AXD also has a **Debugger Internals** window that you can use to access other variables.

4.5.2 Internal variable support by processor

The set of variables that is available depends partly on the processor that is selected. For example, variables relating to the system coprocessor are not available on processors that do not have a system coprocessor.

Table 4-1 on page 4-37, Table 4-2 on page 4-38, and Table 4-3 on page 4-39 describe the variables available for each processor group. For a description of the function and allowed values of the variables see *Internal variable descriptions* on page 4-39.

Variables marked with a delta Δ are not used by the AXD debugger in ADS v1.1 (or a later version). AXD and Multi-ICE Version 2.1 (or later) both support a mechanism that describes the target to the debugger, making these variables unnecessary.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Variables marked with a sigma Σ are incorporated into the AXD properties interface and do not appear in the debugger internals variable list.

Table 4-1 ARM7 family debugger variable support

Variable name	ARM7[a]	ARM7T[b]	Samsung[c]	ARM7xxT[d]	ARM7EJ-S
cp_access_code_address	Yes	Yes	Yes	Yes	No
cp15_current_memory_area Δ	No	No	No	Yes[e]	No
icebreaker_lockedpoints	Yes	Yes	Yes	Yes	Yes
internal_cache_enabled	Yes	Yes	Yes	Yes	Yes
internal_cache_flush	Yes	Yes	Yes	Yes	Yes
ks32c_special_base_address	No	No	Yes	No	No
safe_non_vector_address	Yes	Yes	Yes	Yes	Yes
semihosting_dcchandler_address Σ[f]	No	Yes	Yes	Yes	Yes
semihosting_enabled =0 or =1 Σ	Yes	Yes	Yes	Yes	Yes
semihosting_enabled =2 Σ	No	Yes	Yes	Yes	Yes
sw_breakpoints_preferred	Yes	Yes	Yes	Yes	Yes
system_reset	Yes	Yes	Yes	Yes	Yes
top_of_memory	Yes	Yes	Yes	Yes	Yes
user_input_bit [1,2]	Yes	Yes	Yes	Yes	Yes
user_output_bit [1,2]	Yes	Yes	Yes	Yes	Yes
vector_address	No	No	No	Yes[g]	Yes[h]

a. ARM7 includes ARM7DI, ARM7DMI, and devices using these cores.
b. ARM7T includes ARM7TDI, ARM7TDMI, ARM7TDI-S, ARM7TDMI-S, and devices using these cores.
c. Samsung includes the KS32C50100 and the S3C4510B.
d. ARM7xxT includes ARM710T™, ARM720T™, and ARM740T™.
e. Only ARM740T.
f. This must be in the range of an ARM branch instruction (approximately ±32MB) from the SWI vector. It must not rely on a negative branch from low memory wrapping around to high memory.
g. Only ARM720T.
h. There is no system coprocessor.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-2 ARM9 family debugger variable support

Variable name	ARM9T[a]	ARM9xxT[b]	ARM9E-S	ARM9xxE-S[c]
cp_access_code_address	No	Yes	No	Yes
cp15_cache_selected Δ	No	Yes	No	Yes
cp15_current_memory_area Δ	No	Yes[d]	No	Yes[e]
icebreaker_lockedpoints	Yes	Yes	Yes	Yes
internal_cache_enabled	Yes	Yes	Yes	Yes
internal_cache_flush	Yes	Yes	Yes	Yes
safe_non_vector_address	Yes	Yes	Yes	Yes
semihosting_dcchandler_address Σ[f]	Yes	Yes	Yes	Yes
semihosting_enabled Σ	Yes	Yes	Yes	Yes
sw_breakpoints_preferred	Yes	Yes	Yes	Yes
system_reset	Yes	Yes	Yes	Yes
top_of_memory	Yes	Yes	Yes	Yes
user_input_bit [1,2]	Yes	Yes	Yes	Yes
user_output_bit [1,2]	Yes	Yes	Yes	Yes
vector_address	No	Yes[g]	Yes[h]	Yes

a. ARM9T includes ARM9TDMI™ and devices using the cores.
b. ARM9xxT includes ARM920T, ARM922T™, ARM925T™, and ARM940T.
c. Includes ARM926EJ-S, ARM946E-S, and ARM966E-S.
d. Only ARM940T.
e. Only ARM946E-S.
f. This must be in the range of an ARM branch instruction (approximately ±32MB) from the SWI vector. It must not rely on a negative branch from low memory wrapping around to high memory.
g. Excludes ARM940T (Rev 0).
h. There is no system coprocessor.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-3 ARM10 family and XScale microarchitecture debugger variable support

Variable name	ARM10[a]	XScale
cp15_cache_selected Δ	Yes	No
icebreaker_lockedpoints	No	No
internal_cache_enabled	Yes	Yes
internal_cache_flush	Yes	Yes
safe_non_vector_address	No	No
semihosting_dcchandler_address Σ[b]	Yes	No
semihosting_enabled Σ	Yes	Yes
sw_breakpoints_preferred	Yes	Yes
system_reset	Yes	Yes
top_of_memory	Yes	Yes
user_input_bit [1,2]	Yes	Yes
user_output_bit [1,2]	Yes	Yes
vector_address	Yes	Yes

a. Includes ARM1020T, and ARM10200T.
b. This must be in the range of an ARM branch instruction (approximately ±32MB) from the SWI vector. It must not rely on a negative branch from low memory wrapping around to high memory.

4.5.3 Internal variable descriptions

This is a list of the debugger internal variables that Multi-ICE makes available to you through the debugger. Refer to the manual for your debugger for information on reading and writing them:

`cp_access_code_address`

This specifies an area of memory, of at least 40 bytes, that can be used by Multi-ICE during read or write coprocessor operations. Multi-ICE ensures that this memory is reloaded with its original values after use. This area of memory must be readable, writable, and executable.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`cp15_cache_selected`

This is only valid on Harvard Architecture processors. This includes the ARM9 and ARM10 families, but excludes the ARM7 family. It is not valid with XScale processors.

Note

If you are using AXD, this variable is not available. Instead, Multi-ICE describes the target coprocessor register names and access information to AXD. AXD includes the described registers in the processor register view, enabling you to read and modify the values as required.

It indicates the alias of the CP15 register that is read/written. Use one of the values from Table 4-4.

Table 4-4 Cache selection type values

Value	CPU type	Description
0	ARM9 or ARM10 (Harvard) cores	Select the *Data Cache* (DCache)
1	ARM9 or ARM10 (Harvard) cores	Select the *Instruction Cache* (ICache)
2	ARM946E-S, ARM966E-S	Select tightly-coupled instruction memory
3	ARM946E-S, ARM966E-S	Select tightly-coupled data memory

For further information on the CP15 registers, refer to Appendix E *CP15 Register Mapping*, and to the ARM technical reference manual for the processor being used.

`cp15_current_memory_area (0-7=Memory areas 0-7)`

This selects the memory area to be accessed in register 6 on processors that support multiple protection regions. If necessary, the `cp15_cache_selected` variable selects between a data or instruction memory area.

For further information on the CP15 registers, refer to Appendix E *CP15 Register Mapping*, and to the ARM technical reference manual for the processor being used.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

If you are using AXD, this variable is not available. Instead, Multi-ICE describes the target coprocessor register names and access information to AXD. AXD includes the described registers in the processor register view, enabling you to read and modify the values as required.

`icebreaker_lockedpoints`

This variable controls user access to the EmbeddedICE logic watchpoint registers. It is a bitmask, with bit 0 relating to watchpoint unit 0 and bit 1 relating to watchpoint unit 1. If an IEU is built into the processor, then bit 2 relates to IEU unit 0, bit 3 to IEU unit 1, up to bit 31 relating to IEU unit 29.

If a bit in the bitmask is set (1) then Multi-ICE does not use the related watchpoint unit. If it is unset (0), then Multi-ICE can use the unit. Refer to *Using the EmbeddedICE logic values* on page 4-69 for more information.

`internal_cache_enabled (0=disabled, 1=enabled)`

This variable controls the behavior of the memory cache within the Multi-ICE DLL. On AXD startup, it has the same value as the **Cache Enabled** flag in the Multi-ICE Advanced Settings window. The user can then change this value to override the initial value, and so enable or disable the cache.

`internal_cache_flush`

This variable always reads as 0. Writing a nonzero value to it causes the Multi-ICE internal memory cache to be flushed.

`ks32c_special_base_address`

This variable contains the address of the special system registers in the Samsung KS32C50100 or Samsung S3C4510B processor. These registers can be mapped to one of many pages of memory, and it is not possible to ask the device where they are. One register from this bank is required by the Multi-ICE cache manipulation code.

`safe_non_vector_address`

This variable defaults to `0x10000`. This variable must be set to the base address of a 64KB area of memory that does not overlap the 64KB block of memory starting at `vector_address`. The block of memory that is referenced must be *safe*, in that the Multi-ICE DLL might cause some

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

reads from this area to occur, and these reads must be harmless. Memory reads must not cause Data Aborts and must not affect I/O devices. Multi-ICE does not write to this memory area.

`semihosting_dcchandler_address`

When the value of `semihosting_enabled` is 2, the value of `semihosting_dcchandler_address` is the address of the SWI handler. See *Semihosting* on page 4-50.

`semihosting_enabled`

This variable controls the semihosting facility in the Multi-ICE DLL. See *Semihosting* on page 4-50.

`sw_breakpoints_preferred`

This variable controls the breakpoint selection algorithm. If it is nonzero, the breakpoint selection algorithm chooses to use software breakpoints wherever possible (for example, it does not set software breakpoints in ROM).

If it is zero, it chooses to use hardware breakpoints unless the number of breakpoints required exceeds the number of hardware breakpoint units available. See Appendix B *Breakpoint Selection Algorithm* for more information.

`system_reset`

When read, this is always zero. If written with a nonzero value the target board is immediately reset using system reset pulse of approximately 250ms.

`top_of_memory`

This variable defines the highest address in memory that the C-library uses for stack space. The value is transferred to the target in the result of the `SYS_HEAPINFO` semihosting SWI call. The default value is `0x80000`, meaning that the first word pushed to the stack is written to `0x7FFFC`.

For the purposes of `SYS_HEAPINFO`, Multi-ICE assumes the memory map shown in Figure 4-21 on page 4-43.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

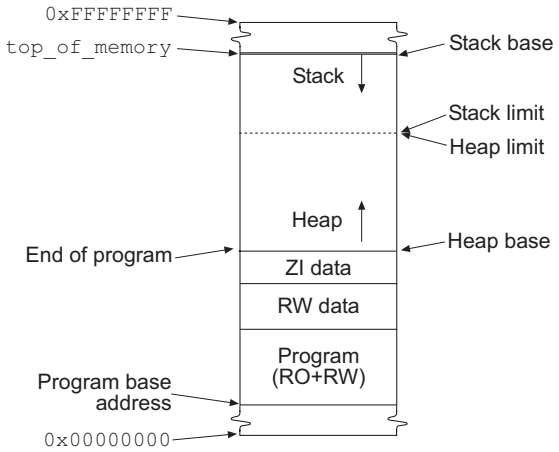


Figure 4-21 Relating top_of_memory to single section program layout

Note

- If the application is scatterloaded the application must include a user-defined function (`__user_initial_stackheap`) that defines the stack and heap limits. Therefore, if the application does not call `SYS_HEAPINFO` explicitly, the target ignores the value of `top_of_memory`.
- The value of `top_of_memory` must be higher than the sum of the program base address and program size. If set incorrectly, the program might crash because of stack corruption or because the program overwrites its own code.
- There is no requirement that `top_of_memory` is at the true top of memory. A C or assembler program can use memory at higher addresses.

`user_input_bit1`, `user_input_bit2`

These variables show the state of the two user input bits. These are not polled, so they show the state at the time the **Debugger Internals** window was displayed, the core was last stopped, or the command-line command was executed.

`user_output_bit1`, `user_output_bit2`

These variables allow you to alter the state of the user output bits. You can only change the output bits if they are assigned to this connection, and if the **Set by Driver** option is enabled. These are set on the server using **User Output Bits**, found under the **Settings** menu (see *User output bits*

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dialog on page 3-19). These are not polled, so they show the state at the time the Debugger Internals window was displayed or the command-line command was executed.

`vector_address`

This variable applies only to processors that support movable vector tables, such as ARM720T™ and ARM920T. It tells Multi-ICE where the exception vector table is. The default value is the vector address that was current the last time the processor stopped. The address is determined by reading the V bit from CP15 Register 1. You can set it to either `0` or `0xFFFF0000`. There must be readable memory at this address.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.6 Post-mortem debugging

This section describes how to examine the state of a system that has previously been running but that is currently not connected to Multi-ICE. For example, you can find out why a program has stopped.

Before you can examine a running target with Multi-ICE, you must configure the Multi-ICE interface unit and the server for that target. If you have a target that is operating without a Multi-ICE interface unit connected, and you must examine it to find out why it is behaving in a particular way, you must power up Multi-ICE interface unit and configure the server without disturbing the state of the target. This requires that the Multi-ICE interface unit is powered before it is connected to the target. The possible ways of powering Multi-ICE are described in:

- *Powering the interface unit using the power jack (Multi-ICE Version 2.1 or later)*
- *Powering the interface unit using a modified cable* on page 4-46
- *Powering the interface unit using the 14-way JTAG adaptor, HPI-0027* on page 4-47.

4.6.1 Powering the interface unit using the power jack (Multi-ICE Version 2.1 or later)

From Version 2.1 onwards, the Multi-ICE interface hardware supplied with Multi-ICE includes:

- a power input jack socket next to the JTAG connector
- power conditioning and switching circuit that enables you to plug and unplug the JTAG cable without affecting the target.

The power jack accepts a 2.1mm plug with center positive polarity. You can use power supplies with output voltages from 9V to 12V DC and a minimum continuous power rating of 500mA.

Note

The voltage reference used by the interface unit JTAG circuit is generated from the **VTref** signal present on the JTAG connector. If this signal is not connected at the target, you must modify the target or the JTAG cable to supply a suitable reference. Typically, connecting **VTref** to **Vsupply** is sufficient.

To connect to a running target:

1. Ensure that the Multi-ICE interface unit is not already powered from the target.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. The JTAG input lines **TDI**, **TMS**, **nSRST**, and **nTRST** must have pull-up resistors (normal practice) and **TCK** must have a pull-down resistor so that when the adaptor is not connected from the target these lines are in their quiescent state.
3. Plug the power jack into the interface unit.
4. Configure the Multi-ICE server. You must either manually configure the server or autoconfigure using a separate test system. Leave the server running.

   Note

   Do not use autoconfigure on the target that must be debugged. Doing so resets the processor.
5. Plug the 20-way JTAG cable into the target.
6. Start the debugger. The debugger can then stop the processor and display the stopped state.

   To get a high-level (source code) view of the problem, you must load the symbol table for your target program into the debugger. For AXD, use **File → Load debug symbols...**. For ADW or ADU, use **File → Load symbols only...**.
7. Press the **Go** or **Run** button and unplug the JTAG connector to restart the system. Then exit the debugger.

4.6.2 Powering the interface unit using a modified cable

Note

Use this method if you are not using the Multi-ICE Version 2.1 (or later) interface unit, and you have a 20-way JTAG connector on the target.

To connect to a running target with a modified cable:

1. Ensure that the Multi-ICE interface unit is not already powered from the target.
2. You must make a special IDC cable to split off pin 1 (**VTref**), pin 2 (**Vsupply**) and **GND**.
3. Connect a 5V power supply between **Vsupply** and **GND** to power the Multi-ICE interface unit, but do not switch it on.
4. Connect a voltage suitable for your target to pin 1 of the 20-way Multi-ICE connector (**VTref**) through a 1kΩ resistor. Only a very small power supply current is required into **VTref**. You can use the same power supply for this reference

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

voltage as that used to power the Multi-ICE interface unit if your target uses 3.3V or 5V logic. If your target runs at a lower voltage, you must apply that voltage to **VTref** (see Chapter 6 *System Design Guidelines* for more details about **VTref**).

5. The JTAG input lines **TDI**, **TMS**, **nSRST**, and **nTRST** must have pull-up resistors (normal practice) and **TCK** must have a pull-down resistor so that when the cable is disconnected from the target these lines are in their quiescent state.

6. Switch on the power between **Vsupply** and **GND**, and between **VTref** and **GND**. The power light on the Multi-ICE interface unit glows brightly.

7. Configure the Multi-ICE server. You must either manually configure the server or autoconfigure using a separate test system. Leave the server running.

   Note

   Do not use autoconfigure on the target that must be debugged. Doing so resets the processor.

8. Plug the 20-way JTAG cable into the target.

9. Start the debugger. The debugger can then stop the processor and display the stopped state.

   To get a high-level (source code) view, you must load the symbol table for your target program into the debugger. For AXD, use **File → Load debug symbols...**. For ADW or ADU, use **File → Load symbols only...**.

10. Press the **Go** or **Run** button and unplug the JTAG connector to restart the system. Then exit the debugger.

4.6.3 Powering the interface unit using the 14-way JTAG adaptor, HPI-0027

To connect to a running target using the optional 14-way adaptor:

1. Ensure that the Multi-ICE interface unit is not already powered from the target.

2. You must remove the link on connector J3 of the 14-way adaptor and connect a 5V power supply to the +5V and 0V pins. The **VTref** signal is permanently connected to the target power supply.

3. The JTAG input lines **TDI**, **TMS**, **nSRST**, and **nTRST** must have pull-up resistors (normal practice) and **TCK** must have a pull-down resistor so that when the adaptor is disconnected from the target these lines are in their quiescent state.

4. Switch on the power between **Vsupply** and **GND**. The power light on the Multi-ICE interface unit glows brightly.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5. Configure the Multi-ICE server. You must either manually configure the server or autoconfigure using a separate test system. Leave the server running.

   Note

   Do not use autoconfigure on the target that must be debugged. Doing so resets the processor.

6. Plug the 20-way JTAG cable into the target.

7. Start the debugger. The debugger can then stop the processor and display the stopped state.

   To get a high-level (source code) view of the problem, you must load the symbol table for your target program into the debugger. For AXD, use **File → Load debug symbols...**. For ADW or ADU, use **File → Load symbols only...**.

8. Press the **Go** or **Run** button and unplug the JTAG connector to restart the system. Then exit the debugger.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.7 Access to CP15

Multi-ICE provides support for coprocessors. Part of the description of the ARM processors includes a description of the system control coprocessor, CP15, so you do not have to describe them to the debugger.

For a general description of the ARM coprocessor architecture and the general form of CP15, see the *ARM Architecture Reference Manual*. For information on the precise facilities offered by your processor, see the processor technical reference manual. Many of these are available in PDF form from the ARM website and on paper on request. Additional information is available in Appendix E *CP15 Register Mapping*.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.8 Semihosting

Semihosting enables the ARM processor target to make I/O requests to the computer running the debugger. This means the target does not require a screen, keyboard, or disk during the development period. These requests are made as a result of calls to C library functions, for example, `printf()` and `getenv()`. Semihosting using Multi-ICE is described in the following sections:

- *Enabling semihosting*
- *Adding an application SWI handler when using Multi-ICE* on page 4-52.

4.8.1 Enabling semihosting

When using the Multi-ICE DLL, semihosting is handled with either a real `SWI` exception handler, or by emulating a handler using breakpoints. You can modify this semihosting mechanism using the following debugger internal variables:

`semihosting_enabled`

By default, this variable is set to 1 to enable breakpoint semihosting but you can set it to the following values:

0 Disables semihosting.

1 Enables start-stop semihosting, using breakpoint-based emulation of the SWI handler.

2 Enables DCC semihosting, using an exception handler that uses DCC to communicate with the host.

The `S` bit in `vector_catch` must not be used as an alternative to changing `semihosting_enabled`.

`semihosting_vector`

This variable controls the location of the breakpoint set by the Multi-ICE DLL to detect a semihosted `SWI`. It is set to 8 by default unless `vector_address` specifies high vectors are in use.

In ADW and ADU, you can access both of these variables by selecting **Debugger Internals** from the **View** menu. In AXD, debugger internal variables are accessed with dedicated windows. See the *ADS Debuggers Guide* for more information.

Start-stop semihosting

Start-stop, or standard, semihosting involves setting a breakpoint either on the SWI vector or somewhere else in cooperation with your own SWI handler, depending on the value of `semihosting_vector`.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When the breakpoint is hit Multi-ICE interprets it as a semihosting request:

- the processor registers and memory are read as required to decode the request
- the request is executed on the host
- the return value is placed in register R0 and, when required, memory is modified
- the pc is modified so the next instruction is the instruction following the SWI
- execution is resumed.

Note

Using Multi-ICE standard semihosting with systems that include time-sensitive interrupt-driven software is not recommended. The processor must be halted while a semihosting operation is performed, and so interrupts will be missed. Use *DCC semihosting* or ARM RealMonitor to debug these systems

The breakpoint on the SWI vector uses breakpoint resources that might be required for other purposes.

DCC semihosting

DCC semihosting offers two advantages to standard breakpoint-based semihosting:

- it is in most cases faster
- it does not cause the target processor to enter debug state and so interrupts continue to be serviced.

Standard semihosting is the initial semihosting mode because DCC semihosting is more intrusive on the target.

Because DCC semihosting does not cause the processor to halt, this method of semihosting is more suitable for real-time systems. It is also more useful for targets that use two or more processors in a JTAG chain, because DCC semihosting does not interfere with automatic starting and stopping of processors.

You cannot use the DCC for other purposes (for example, Channel Viewers) while DCC semihosting is enabled.

The DCC semihosting SWI handler is installed in target memory at the address in the variable `semihosting_dcchandler_address`. It is vital that:

- The address is in the range of an ARM branch instruction (approximately ±32MB) from the SWI vector. It must not rely on a negative branch from low memory wrapping around to high memory.
- The memory that the debugger writes the handler to is unused.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The SWI handler is no more than 0.75KB in size and is written to memory whenever either:

- DCC semihosting is enabled by setting `semihosting_enabled` to two
- `semihosting_dcchandler_address` is changed and DCC semihosting is already enabled.

The default value for `semihosting_dcchandler_address` is `0x70000`. To change the location of the handler, you must:

1. Disable semihosting by setting `semihosting_enabled` to zero.
2. Change the address of the handler by setting the variable `semihosting_dcc-handler_address` to a new value.
3. Enable DCC semihosting by setting `semihosting_enabled` to two.

Note

With processors that use Rev C or earlier AMBA wrappers you cannot use DCC-hosted semihosting (`semihosting_enabled=2`). Use `semihosting_enabled=1` (stop/start semihosting) instead.

4.8.2 Adding an application SWI handler when using Multi-ICE

Many applications require their own SWI handlers as well as using semihosting SWIs. You must do this so that the application SWI handler cooperates with the Multi-ICE semihosting mechanism as follows:

1. Install the application SWI handler into the vector table.
2. For standard semihosting, modify the value of `semihosting_vector` to point to a location that is only reached if your handler does not recognize the SWI, or recognizes it as a semihosting SWI.
3. For DCC semihosting, install the application SWI handler on the SWI vector. When a SWI the application does not recognize occurs, branch to `semihosting_dcchandler_address+12` with the processor state as it was on entry to the SWI handler. The DCC semihosting handler executes the request and returns to the calling code.

For example, a particular SWI handler might detect if it has failed to handle a SWI and branch to an error handler (see the ADS documentation for further details of writing SWI handlers). An example of a basic exception handler is shown in Example 4-3 on page 4-53.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 4-3 Basic SWI handler

```
                            ; r0 = 1 if SWI handled
    CMP r0, #1              ; Test if SWI has been handled.
    BNE NoSuchSWI           ; Call unknown SWI handler.
    LDMFD sp!, {r0}         ; Unstack SPSR...
    MSR spsr_cf, r0         ; ...and restore it.
    LDMFD sp!, {r0-r12, pc}^ ; Restore registers and return.
```

You can modify this code for use in conjunction with Multi-ICE start-stop semihosting as shown in Example 4-4.

Example 4-4 SWI handler with Multi-ICE link

```
                           ; r0 = 1 if SWI handled
    CMP r0, #1             ; Test if SWI has been handled.
    LDMFD sp!, {r0}        ; Unstack SPSR...
    MSR spsr_cf, r0        ; ...and restore it.
    LDMFD sp!, {r0-r12, lr} ; Restore registers.
    MOVEQS pc, lr          ; Return if SWI handled.
Semi_SWI
    MOVS pc, lr
```

You must then set up the `semihosting_vector` with the address of `Semi_SWI`. The instruction at this address is never actually executed because the Multi-ICE DLL returns directly to the application after processing the semihosted SWI. Using a normal SWI return instruction ensures that the application does not crash if the semihosting breakpoint is not set up.

If the application is linked with the semihosted ARM C library, and therefore uses the C library startup code, you must change the contents of `semihosting_vector` just before the application installs its own handler, typically by setting a breakpoint in the main code. This is because, if `semihosting_vector` is set to the fall-through part of the application SWI handler before the application starts execution, the semihosted SWIs that are called by the library initialization can trigger an unknown watchpoint error. At this point, the SWI vector has not yet had the application handler written to it, and might still contain the software breakpoint bit pattern. This triggers a breakpoint that the Multi-ICE DLL does not know about because the `semihosting_vector` address has moved to a place that cannot currently be reached.

Note

If semihosting is not required by your application, including the startup code, you can simplify this process by setting `semihosting_enabled` to zero.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You must take care when moving an application that previously ran in conjunction with the Angel debug monitor onto a Multi-ICE system. On Angel debug monitor systems, application SWI handlers are typically added by moving and adjusting the contents of the Angel installed SWI vector to another place, and installing the application SWI handler into the SWI vector. This method does not apply to the Multi-ICE DLL because there is no instruction to move out of the SWI vector, and no code to jump to. Therefore, when moving an application onto a Multi-ICE based system, you must convert to the Multi-ICE way of installing the application and semihosted SWI handlers.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.9 Watchpoints and breakpoints

The ARM debuggers provide break and watch facilities with Multi-ICE targets. The following sections describe the facilities that are implemented and what the implications for you are:

- *Watchpoints*
- *Breakpoints* on page 4-56
- *Watchpoints, breakpoints, and the program counter* on page 4-56
- *Vector breakpoints and exceptions* on page 4-57
- *Vector catch with ROM at 0x0* on page 4-58
- *Stopping the processor* on page 4-58.

Refer to Appendix B *Breakpoint Selection Algorithm* for more information on how the breakpoint hardware is managed.

4.9.1 Watchpoints

All ARM debugger watchpoints are data-changed watchpoints. That is, they are not activated if the data point is read or written to with the same data value as the one currently in memory. See *Accessing the EmbeddedICE logic directly* on page 4-65 for details of how to implement other forms of watchpoint.

Hardware versus software watchpoints

Hardware watchpoints are implemented using an EmbeddedICE logic point to detect data writes to addresses that fall inside a mask. This type of watchpoint is efficient because execution stops only when the relevant data is written. However, it completely ties up an EmbeddedICE logic point.

Note

If a structure or an array is being watchpointed, the mask is likely to include some addresses that are not part of the object being watchpointed. In this case, writes to these unwanted addresses are filtered out by the debugger. Execution performance is slightly degraded because the processor is stopped when the unwanted watchpoint is hit, and then restarted automatically by the debugger.

Software watchpoints use the EmbeddedICE logic differently. Instead, after each instruction is executed, the data locations concerned are examined to see whether their values have changed. If a value has changed, execution is halted. Otherwise, execution is restarted. This type of watchpoint significantly reduces execution performance. In addition, it cannot be used on write-only areas of memory, such as some types of memory-mapped device register.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.9.2 Breakpoints

When you inspect the current breakpoints and watchpoints (using the watch or break commands without arguments in the ADW command window, or by viewing the **Breakpoints** or **Watchpoints** windows in ADW, AXD, or ADU), the output specifies whether they are hardware or software breakpoints or watchpoints.

Hardware versus software breakpoints

Hardware breakpoints are implemented using an EmbeddedICE logic point to detect an instruction fetch from the appropriate address. This works in all cases, even if the program being debugged modifies itself as it executes, or if the code is in ROM. However, it completely ties up one of the two available EmbeddedICE logic point units.

For ARM processors prior to ARM architecture v5, software breakpoints are implemented using an EmbeddedICE logic unit to detect an instruction fetch of a particular bit pattern. This bit pattern has been stored previously at the appropriate location, and the real instruction stored in the host debugger memory. Any number of software breakpoints can be supported using a single EmbeddedICE logic point.

ARM architecture v5 processors have specific breakpoint instructions, so extra EmbeddedICE logic is not required.

Self-modifying code, code in ROM, or code paged from disk file cannot be debugged using software breakpoints. If you attempt to set a breakpoint on a location in ROM, Multi-ICE detects that the memory is not writable and tries to use a hardware breakpoint.

4.9.3 Watchpoints, breakpoints, and the program counter

Watchpoints are taken when the data being watchpointed has changed. When this happens, the program counter is updated to point to the instruction following the one that caused the watchpoint to be taken. The value of the watchpointed data is therefore the new value, not the old value.

Breakpoints are taken when the instruction being breakpointed reaches the Execute stage of the pipeline, but before it is executed. So, when the breakpoint is taken, the program counter is not updated, and retains the address of the breakpointed instruction.

Note

Inside the core of an ARM CPU, the program counter typically points to two instructions beyond the currently executing instruction. (Historically, this is the address of the instruction currently being loaded into the Fetch stage of the pipeline.) The ARM

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

debuggers simplify this by reporting a modified value for the program counter so that when it is displayed within the debugger its contents are the address of the instruction being, or about to be, executed.

4.9.4 EmbeddedICE/RT breakpoints

The EmbeddedICE/RT logic is an upgrade of the EmbeddedICE logic. It is included in ARM7TDMI processor cores from Rev 4 onwards and ARM9TDMI processor cores from Rev 2 onwards, and provides support for real time debugging. Many of the improvements are only useful to a target based monitor, such as RealMonitor. Multi-ICE can, however, use the RT extensions to set breakpoints and watchpoints while the target program is running. This is only possible when the target debugger also supports this capability. The version of AXD included with ADS v1.1 or later does include this support.

4.9.5 Vector breakpoints and exceptions

The Multi-ICE DLL puts into place any breakpoints that have been requested in the order they are received, including those implicitly requested by the debugger internal variable `vector_catch`. The Multi-ICE DLL uses the breakpoint resources it has available as efficiently as possible, but the presence of a vector catch breakpoint sometimes requires software breakpoints to be used.

The `vector_catch` variable indicates whether or not execution must be trapped when one of the conditions described in Table 4-5 arises. The default value is, for ADW `%RUsPDAifE`, and for AXD `%RUsPDif` where capital letters indicate that the condition is to be intercepted.

Table 4-5 Breakpoints

Vector	Description
R	Reset
U	Undefined instruction
S	Software interrupt (SWI)
P	Instruction prefetch abort
D	Data access abort
A[a]	Address exception

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-5 Breakpoints (continued)

Vector	Description
I	Interrupt request (IRQ)
F	Fast interrupt request (FIQ)
E[b]	Error

a. Not used by AXD.
b. Not used by AXD or ADW.

On ARM9TDMI and ARM10TDMI family devices, and on XScale microarchitecture processors, additional hardware in the core enables you to perform vector catching without setting general breakpoints on the vectors. See the *ARM9TDMI Technical Reference Manual* or *ARM1020T Technical Reference Manual* for more details.

In normal usage, the `SWI` flag within `vector_catch` remains lowercase, as finer control is provided by the debugger internal variables `semihosting_enabled` and `semihosting_vector` (see *Semihosting* on page 4-50).

Note

If you set the `S` bit in `vector_catch` with `semihosting_enabled` nonzero, vector catch takes precedence over semihosting.

4.9.6 Vector catch with ROM at 0x0

In systems where there is ROM at address `0x0`, you must take care with the setting of `vector_catch`. See *Debugging applications in ROM* on page 4-62 for more details.

4.9.7 Stopping the processor

There are two ways to stop an ARM processor:

- Asserting **DBGRQ** and waiting for **DBGACK**. This is the standard method.
- Asserting **nSRST** and pulsing **nTRST**, setting a breakpoint on the Reset vector, and then releasing **nSRST**. This is the alternative method.

If the standard method fails, then you are asked whether the alternative method can be tried. This resets both the core and TAP controller and leaves the TAP controller operating. It then programs a hardware breakpoint on address 0, releases system reset, and waits to see if the processor hits the breakpoint.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The reset method is useful if the core is not being clocked, for example if **nWait** is permanently asserted, because it allows the debugger to regain control of the processor.

Note

This alternative method is more intrusive than the standard method, because it resets all processors in a multi-processor system.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.10 Cached data

The way that Multi-ICE handles cached data depends on the type of processor that is being debugged:

- If you are debugging a processor with an ARM7, ARM9, or ARM10 core, see *Cached data on ARM architecture processors*.
- If you are debugging an XScale microarchitecture processor, see *Cached data on XScale microarchitecture processors* on page 4-61.

4.10.1 Cached data on ARM architecture processors

When debugging a cached processor with an ARM7, ARM9, or ARM10 core, Multi-ICE preserves as much of the cache contents as possible. In the ideal case, it uses the following strategy:

- When the processor enters debug state with caches enabled, Multi-ICE does the following:
  - — it disables line fills to both the ICache and the DCache
  - — it stores the current write behavior for the DCache.
  - — it selects write-through behavior for the DCache.
- If data is read from cachable memory in debug state, Multi-ICE does not read it into the cache, because line fills have been disabled.
- If data is written to memory in debug state, Multi-ICE does the following:
  1. It invalidates all cache lines in the ICache that contain an address within the data.

     All other cache lines in the ICache remain unaffected.
  2. It update the data values in all cache lines in the DCache that contain an address within the data. It writes the data through into memory, because of the write-through behavior.

     All other cache lines in the DCache remain unaffected.

     Note

     If an address within the data is not in the cache prior to the write, it is not added to the cache, because line fills have been disabled.
- When the processor leaves debug state, Multi-ICE does the following:
  - — it re-enables line fills to both the ICache and the D cache
  - — it restores the stored write behavior for the DCache.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

The effectiveness of cache preservation depends on the exact processor and revision being used. In some cases, limitations in the design of the processor prevent Multi-ICE from using this ideal cache preservation strategy. However, it always ensures that no data is lost, and that cache coherency is maintained.

Locked-Down Data

If you invalidate a cache line that is in a lockdown block, any dirty data in the cache line is lost. The lock down remains in effect. Because the cache line has been invalidated, no further cache hits occur for that line, and so that cache line remains unused, even after exiting debug state. Any subsequent accesses to the invalidated addresses instead use the unlocked region of the cache.

4.10.2 Cached data on XScale microarchitecture processors

When debugging a cached XScale microarchitecture processor, Multi-ICE uses the following strategy:

- When the processor enters debug state with the DCache enabled, Multi-ICE does the following:
  - it cleans the entire DCache, and so flushes its contents (but see *Locked-Down Data*).
- If data is read from cachable memory in debug state, Multi-ICE reads it into the cache, because line fills have not been disabled.
- If data is written to memory in debug state, Multi-ICE does the following:
  1. It flushes the entire ICache.
  2. It update the data values in all cache lines in the DCache that contain an address within the data. It writes the data through into memory, simulating write-through behavior.

     All other cache lines in the DCache remain unaffected.

Locked-Down Data

All XScale microarchitecture processors protect lockdown blocks from cache clean and invalidate operations. So on entry to debug state, cache lines that are in a lockdown block are not cleaned and flushed. If you subsequently disable the cache through the debugger, data written to locked cache regions might be lost.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.11 Debugging applications in ROM

This section describes some of the issues involved with debugging applications in ROM using Multi-ICE in the following sections:

- *Debugging from reset*
- *Debugging systems with ROM at zero* on page 4-64.

4.11.1 Debugging from reset

You can use the Multi-ICE DLL to debug systems running in ROM. Typically, when a target board with an application stored in ROM is powered up, the application begins running. Therefore, when the debugger is started up on the host, the processor on the target is stopped. At this stage, the application can be at any point in its execution lifetime, depending on when the debugger was started.

This means that you can examine the state of the system and restart execution from the current place. In some cases, this is sufficient. However in many cases it is preferable to restart execution of the application as if from power-on. There are two ways to do this:

- *Simulating a reset*
- *Carrying out a real reset* on page 4-63.

When you debug code running from ROM, ensure that at least one watchpoint unit remains available to allow breakpoints to be set on code in ROM, because you cannot use software breakpoints. The chances of the debugger taking these units for its own use can be reduced by not using standard semihosting or vector catching. To do this on a processor without vector catch hardware you must set the following debugger internal variables as soon as possible after starting up the debugger:

```
semihosting_enabled = 0
vector_catch = 0
```

Next set up any ROM breakpoints before any non-ROM breakpoints or watchpoints are set. Otherwise the watchpoint units might be full, causing the attempt to set the ROM breakpoint to fail with a debugger-dependent message saying that there are too many breakpoints already set to add another.

Another factor in debugging a system in ROM is that the ROM image does not contain any debug information. When debugging using the Multi-ICE DLL, symbol or source code information is available by loading the relevant information into the debugger from a file on the host, for example by using **Load Symbols...**.

Simulating a reset

You can often simulate a reset from within the debugger by setting:

- pc to 0 (the address of the reset vector)

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- cpsr to %IF_SVC (to change into Supervisor mode with interrupts disabled).

This simulates the state of the ARM processor at power-on or reset, but it does not allow for a reset memory map or the initialization of any target-specific features such as peripheral registers. You are recommended to modify any of these target-specific features to resemble their startup state before executing the application again, if this is possible. You can automate this procedure with the scripting facilities of your debugger, as shown in Example 4-5 for ADW. The name embed.axf must be replaced with the name of the file that the target is executing. You might also have to change the top_of_memory value shown, depending on the memory layout of your target.

Example 4-5 Suggested reset script for ADS versions of ADW, ADU

```
readsyms embed.axf
pc = 0x0
cpsr = %IFt_SVC
$vector_catch = 0
$semihosting_enabled = 0
$top_of_memory = 0x40000
```

The same example for AXD is shown in Example 4-6.

Example 4-6 Suggested reset script for AXD

```
loadsymbols embed.axf
setpc 0
sreg cpsr 0xd3
spp vector_catch 0
spp semihosting_enabled 0
let $top_of_memory 0x40000
```

Carrying out a real reset

Depending on the design of the reset circuitry, you might be able to carry out a real reset of the board. However, take care about when this is done, because if the EmbeddedICE logic is also reset, the debugger might not be able to regain synchronization. Two forms of reset are required on the board:

- a full power-on reset that resets everything on the board
- a reset button that resets everything on the board except the EmbeddedICE logic.

See Chapter 6 *System Design Guidelines* for more information about the different forms of reset.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If a hardware breakpoint is set on the reset vector (or on the start address of the reset code) and the recommended reset circuit is used, when the target is reset, it halts on reset as required.

Note

If Multi-ICE is unable, because of the wiring of the hardware, to detect a reset on your target, you are recommended to delete all other software breakpoints and watchpoints before resetting the processor.

Example using the ARM Integrator board

The ARM Integrator board implements the required two levels of reset. The reset switch carries out the required initialization reset, so enabling debug from reset. All that is required is to set the hardware breakpoint, and then press the Reset button.

4.11.2 Debugging systems with ROM at zero

When debugging processors without vector catch hardware and with ROM rather than RAM at zero, you must set `vector_catch` to zero. This prevents Multi-ICE from trying to set software breakpoints on the vector table.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.12 Accessing the EmbeddedICE logic directly

To manipulate EmbeddedICE logic registers you use the commands that display and set coprocessor registers, with coprocessor number specified as zero.

Note

- The XScale microarchitecture processors do not have an EmbeddedICE logic block, and on these processors coprocessor zero is a real hardware coprocessor.
- The EmbeddedICE logic coprocessor is not emulated for the ARM10 processor because the EmbeddedICE logic is different.

The following sections describe how to access and use the EmbeddedICE logic registers from a debugger:

- *Reading EmbeddedICE logic registers from AXD*
- *Reading EmbeddedICE logic registers from ADW* on page 4-67
- *Using the EmbeddedICE logic values* on page 4-69
- *Support for the ICE Extension Unit* on page 4-69.

See also the description of the `cwrite` or `setreg` command in the documentation for your ADS debugger.

4.12.1 Reading EmbeddedICE logic registers from AXD

To access the coprocessor zero registers using the AXD GUI, select **Processor Views → Registers**. This displays the registers window, from which you can select the coprocessor zero registers as shown in Figure 4-22 on page 4-66.

Note

If you are using ADS v1.0.1 the EmbeddedICE logic registers in the window are not individually named, and appear under the heading `CoProc 0`.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

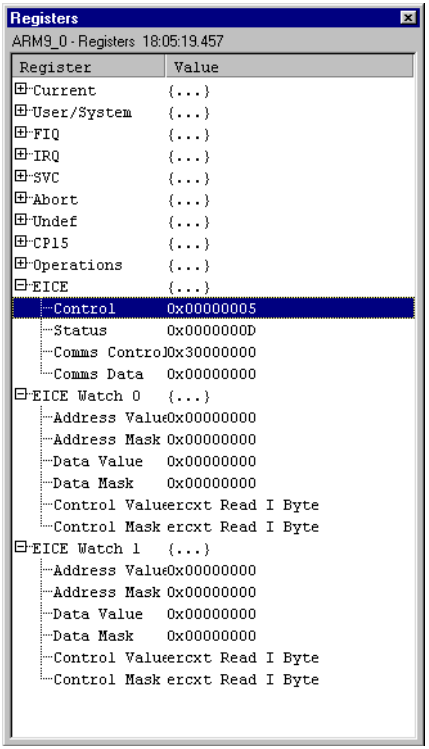


Figure 4-22 Register view showing EmbeddedICE logic registers

The AXD command line interface command `reg` displays the registers in a named group. The bank name varies between different versions of AXD:

ADS 1.0.1 The name is `Coproc 0`

ADS 1.1 or later The name is `EICE`

For example, with ADS 1.2, the command `registers EICE` displays the contents of the EmbeddedICE logic registers, as shown in Example 4-7.

Example 4-7 Displaying coprocessor 0 registers using AXD

```
Debug >reg EICE
Registers Bank: EICE
Index   Name    Value
#1      c0      0x05
#2      c1      0x0D
#3      c2      0x1F
```


Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#4      c4      0x00
#5      c5      0x703008AD
#6      c8      0x00000000
#7      c9      0x00000000
#8      c10     0x00000000
#9      c11     0x00000000
#10     c12     0x0000
#11     c13     0x00
#12     c16     0x00000000
#13     c17     0x00000000
#14     c18     0x00000000
#15     c19     0x00000000
#16     c20     0x0000
#17     c21     0x00
```

4.12.2 Reading EmbeddedICE logic registers from ADW

To access the coprocessor zero registers using the ADW GUI, select **View → Registers** as shown in Figure 4-23.

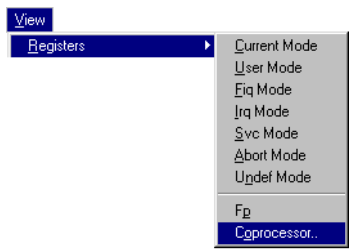


Figure 4-23 The View Registers menu

Display the Coprocessor dialog box. Enter `0` for **Co-processor Number** and check the **Raw (Unformatted)** display option as shown in Figure 4-24.

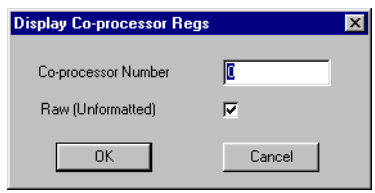


Figure 4-24 The Display Co-processor Regs dialog

Select **OK** to display the coprocessor registers window shown in Figure 4-25 on page 4-68.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

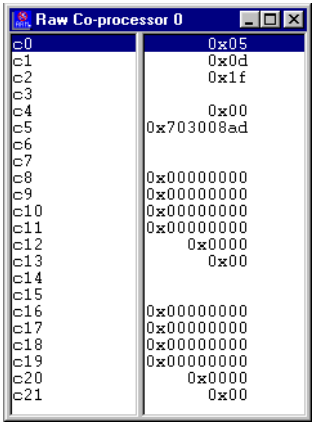


Figure 4-25 EmbeddedICE logic registers in the Raw Co-processor 0 view

The ADW **Command Window** command `cregisters 0` displays the EmbeddedICE logic registers, as shown in Example 4-8.

Example 4-8 Example coprocessor 0 register values

```
Debug: cregisters 0
c0 = 0x05
c1 = 0x09
c4 = 0x00
c5 = 0x00000000
c8 = 0x516ce8da
c9 = 0xbfdf0ea6
c10 = 0xbff6fd7d
c11 = 0xfbaffbff
c12 = 0x0000
c13 = 0xff
c16 = 0x00000008
c17 = 0x00000003
c18 = 0x7dfeeffb
c19 = 0xffffffff
c20 = 0x0100
c21 = 0xf6
```

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.12.3 Using the EmbeddedICE logic values

The register address field in the EmbeddedICE logic scan chain is the coprocessor zero register number. For more information about the EmbeddedICE logic, refer to one of the ARM technical reference manuals on an ARM core with debug capabilities (for example, ARM7TDI or ARM9TDMI).

You can read EmbeddedICE logic registers freely in this manner, but writing to them requires more care. This is because the Multi-ICE DLL also uses EmbeddedICE logic registers to set up breakpoints and watchpoints. When you write to an EmbeddedICE logic register (for example, using the ADW command `cwrite 0 20 0x44`), the Multi-ICE DLL checks to see if that breakpoint register is in use. If it is, the Multi-ICE DLL attempts to free it by degrading hardware breakpoints to software breakpoints. It then sets a lock on that breakpoint register so that the Multi-ICE DLL makes no further attempt to use it.

You can find out the breakpoints that have been locked by displaying the value of `icebreaker_lockedpoints`. You can also write to this variable to unlock breakpoints. In the ARM7 and ARM9 processor families, the breakpoints are numbered 1 and 2, and bits 1 and 2 in `icebreaker_lockedpoints` indicate their status.

If a breakpoint or watchpoint that has been defined by writing to coprocessor zero is taken, the Multi-ICE DLL halts execution with the report `Unknown Watchpoint`. This indicates the breakpoint was outside the control of the Multi-ICE DLL.

Note

Do not write to EmbeddedICE logic registers 0 and 1, the control and status registers. The Multi-ICE DLL uses these to perform many of its operations and changes you make are likely to be lost or to cause the DLL to malfunction.

Debugger requests to read or write EmbeddedICE logic registers do not necessarily cause the registers to be read or written immediately. This is because, for efficiency, the Multi-ICE software only updates the registers just before execution of the program is resumed.

4.12.4 Support for the ICE Extension Unit

Multi-ICE includes support for the ARM *ICE Extension Unit* (IEU). This is a logic block that can be added to a processor when it is built and that extends the number of breakpoint units available to the debugger.

If this unit is present, it is used automatically. The IEU breakpoint registers are numbered from two to 31. The corresponding `icebreaker_lockedpoints` bits are `0x4` to `0x80000000`.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 5
Troubleshooting

This chapter explains the solutions to some of the problems you might encounter when using Multi-ICE. It is split into the following sections:

- *Troubleshooting* on page 5-2
- *Error messages* on page 5-12.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.1 Troubleshooting

This section describes the following situations:

- *The Multi-ICE server fails to autoconfigure the chip* on page 5-3
- *The debugger reports "Attempt to force the processor to enter debug state failed - execution continues"* on page 5-5
- *The debugger reports "Target processor would not enter debug state when requested. Do you want to try asserting System Reset with a breakpoint on address 0?"* on page 5-5
- *The debugger reports "*** Data abort ***" in the execution window* on page 5-6
- *Random stopping or failure to start the debugger* on page 5-6
- *The debugger reports "Hardware interface timeout"* on page 5-7
- *The debugger reports "Unable to set vector catch breakpoints on exception vectors"* on page 5-8
- *Data aborts or crashes when loading or running applications* on page 5-8
- *DCC semihosting, the channel viewer, or the DCC fails* on page 5-9
- *A program that prints strings seems to load and run, but displays garbled text* on page 5-9
- *A 'C' program including string handling or uses char arrays works on some ARM processors but not on others* on page 5-9
- *When trying to connect Multi-ICE and a logic analyzer to an ARM Integrator board to trace a program, Multi-ICE continually reports "The target is being reset, unable to connect"* on page 5-10
- *My application works using ARMulator but quickly crashes when I use Multi-ICE* on page 5-10
- *Running the Multi-ICE server makes my computer run very slowly* on page 5-11
- *I cannot connect to a Multi-ICE server from the computer that is running it* on page 5-11.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.1.1 The Multi-ICE server fails to autoconfigure the chip

The following can be the cause of this problem:

- *The chip contains devices that are not supported by autoconfiguration*
- *You have a signal fault*
- *The server cannot stop the processor during autoconfiguration* on page 5-4
- *There is insufficient or no power to the Multi-ICE interface unit* on page 5-4
- *There is some other problem with the TAP controller* on page 5-4.

The chip contains devices that are not supported by autoconfiguration

If the autoconfiguration process detects a working TAP controller attached to an unknown device, a TAP box is shown in the server window labeled `UNKNOWN`. The list of supported devices is in the file `proclist.txt`. If autoconfiguration of your device is not supported there are two alternatives:

- Use manual configuration. See *Manual device configuration* on page 3-12.
- Double click the TAP box. A dialog box appears that shows the IR length for the detected TAP controller. If you only have one device in your chip with this IR length (excluding devices that are autoconfigurable), you can tell the server what the device is. To do this add a file called `USERDRV`*n*`.TXT` (where *n* is the IR length) to the Multi-ICE installation directory. The file must contain the name of the device to use on the first and only line in the file. For example, if you have a DSP with an IR length of 4, add a file called `USERDRV4.TXT` that contains the text `DSP`.

You have a signal fault

The most common faults are:

- There is no pull-up resistor on the **nTRST** signal. This signal has an open collector drive from the Multi-ICE interface unit. There is no pull-up in the unit. In the EmbeddedICE interface unit this signal was driven with a push-pull driver so no pull-up was required (although it is specified as being required).
- Signals **nTRST** and **nSRST** are not correctly driven by the target. Details on driving these signals correctly are in Chapter 6 *System Design Guidelines*.
- The **TCK** frequency is too high for the chip. The following circumstances are known to cause this:
  - When using a simulated TAP controller in a Quickturn/IKOS unit. In this case the default frequency of 1MHz might be too high. Try autoconfiguring at 20kHz instead by selecting **File → Autoconfigure at 20KHz**. If this fails, manually configure the server.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

— Using a chip whose **MCLK** setting is much slower than 1MHz. A common example is a telephone pager or similar device that contains power-saving (sleep mode) circuitry. On some boards the software can write to a register to switch the clock frequency between fast, for example 20MHz, and slow, say 25kHz. If the processor **MCLK** is running at 25kHz then autoconfiguration fails. Try autoconfiguring at 20kHz instead. If this fails, manually configure the server.

The default **TCK** frequency for autoconfiguration is 1MHz. For all normally working chips this is not a problem.

The server cannot stop the processor during autoconfiguration

This results in `UNKNOWN` being displayed in the TAP controller box. The most common cause is that the processor has not been reset properly. Press the reset button and retry autoconfiguration. If this fails, check that the processor reset line is not continuously asserted (held LOW). If it is, find and fix this problem, and try autoconfiguring again.

There is insufficient or no power to the Multi-ICE interface unit

Multi-ICE requires between 2V and 5V DC on pin 2 of the 20-way JTAG connector, or between 9V and 12V DC on the external power input jack, and a power source that can deliver sufficient current (see *Power supply* on page 2-12 for more details). If you are using a PID board with an ARM7TDMI header card, resistor R1 must be shorted out in order to deliver this.

When the Multi-ICE interface unit is correctly powered, the Power LED is brightly lit.

There is some other problem with the TAP controller

Other possibilities include:

- The signal **nTDOen** is not correctly implemented. The **nTDOen** signal is used to enable the pad tristate driver for the **TDO** pin.
- The target hardware is driving **nWAIT** permanently LOW, or alternatively, **HWAIT** or **BWAIT** permanently HIGH. These signals gate the processor core clock, preventing the EmbeddedICE logic from taking the core into debug state. If this is happening, it usually indicates a fault in the hardware.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.1.2 The debugger reports “Attempt to force the processor to enter debug state failed - execution continues”

Multi-ICE waits for the target to enter debug state when it asserts the **DBGRQ** signal. If this does not happen within the timeout period this error is reported. The following might be the cause of the problem:

- The JTAG clock frequency is too high for this device or this cable length. Try a lower clock frequency.
- The server has been incorrectly configured manually (incorrect number, type, or order of devices, or incorrect IR length given in `IRlength.arm` file).
- The processor is being held in reset state.
- The processor signal **DBGEN** is held LOW (deasserted), preventing the **DBGRQ** being recognized.
- The processor memory interface signal **nWAIT** (or the equivalent bus signals **BWAIT** and **HWAIT**) are permanently asserted, or there is no processor core clock. The processor only acknowledges **DBGRQ** at the end of an instruction, and so anything that prevents the current instruction terminating also prevents the processor entering debug state.
- One or more of the JTAG signals, most often **TCK**, are not of sufficient quality. This can result from a variety of problems in the design of the PCB or the length and type of wiring.

  If the device can be autoconfigured (see *The Multi-ICE server fails to autoconfigure the chip* on page 5-3) but does not work reliably, there is a problem with JTAG signal quality. If you manually configure the server, you get the error message `Can't stop processor`.

5.1.3 The debugger reports “Target processor would not enter debug state when requested. Do you want to try asserting System Reset with a breakpoint on address 0?”

The debugger has tried to use Multi-ICE to access the processor, for example, to stop it, but when the Multi-ICE DLL tried to do this the processor did not respond as expected. This could be because:

- The JTAG clock frequency is too high for this device or this cable length. Try a lower clock frequency.
- The server has been incorrectly configured manually (incorrect number, type, or order of devices, or incorrect IR length given in `IRlength.arm` file).
- The processor is being held in reset state.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- The processor signal **DBGEN** is held LOW (deasserted), preventing the **DBGRQ** being recognized.
- The processor memory interface signal **nWAIT** (or the equivalent bus signals **BWAIT** and **HWAIT**) are permanently asserted, or there is no processor core clock. The processor only acknowledges **DBGRQ** at the end of an instruction, and so anything that prevents the current instruction terminating also prevents the processor entering debug state.
- One or more of the JTAG signals, most often **TCK**, are not of sufficient quality. This can result from a variety of problems in the design of the PCB or the length and type of wiring.

  If the device can be autoconfigured (see *The Multi-ICE server fails to autoconfigure the chip* on page 5-3) but does not work reliably, there is a problem with JTAG signal quality. If you manually configure the server, you get the error message `Can't stop processor`.

You can often regain control of the processor by placing a breakpoint on location zero and then resetting the processor, but this cannot be done without resetting the whole of the target, including any other processors connected to the JTAG chain. Therefore, Multi-ICE asks if it is acceptable to do this.

Note

This method of gaining access to the processor does not work properly unless the system reset **nSRST** and TAP reset line **nTRST** are connected independently, as described in Chapter 6 *System Design Guidelines*, because Multi-ICE must be able to program the TAP controller while the processor is in reset.

5.1.4 The debugger reports "*** Data abort ***" in the execution window

When the debugger starts up, it can stop the processor and read the current value of the pc. The execution window displays the memory around the address of the program counter. If there is no memory at the location, the memory accesses abort causing the observed display. You must load an image file or set the pc register explicitly to point to executable code.

5.1.5 Random stopping or failure to start the debugger

There can be many causes for random failures, mainly due to hardware timing or logic problems. The most common cause of this behavior is failing to pulse **nTRST** LOW when the ARM processor is reset on power-up. Unless this is done, the EmbeddedICE logic is in an undefined state.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.1.6 The debugger reports “Hardware interface timeout”

The following can cause this problem:

- The target hardware has been disconnected from the Multi-ICE interface unit.
- The Multi-ICE interface unit has been disconnected from the workstation.
- There is insufficient power to the Multi-ICE interface unit.

  Multi-ICE requires between 2V and 5V DC on pin 2 of the 20-way JTAG connector, or between 9V and 12V DC on the external power input jack, and a power source that can deliver sufficient current (see *Power supply* on page 2-12 for more details). If you are using a PID board with an ARM7TDMI header card, resistor R1 must be shorted out to deliver this.
- The server cannot establish contact with the Multi-ICE interface unit, because of problems or incompatibilities with the parallel port on your PC.

  The most common causes for this are as follows:

  — The port that Multi-ICE is using is not mapped to its standard address. This *must* be `0x378` for LPT1, and `0x278` for LPT2.

  The facilities to determine and configure this address are dependent on the hardware and operating system that you are using. For more information, refer to the documentation supplied with your computer.

  Note

  PCI-based ports are typically not mapped to these addresses, and so Multi-ICE is unlikely to work with them.

  — The port that Multi-ICE is using is configured to use an incompatible operational mode.

  Reboot your computer, and change the BIOS settings for the parallel port from **ECP** or **EPP** to **8-bit bidirectional** (sometimes called **Standard** or **Normal**). If this succeeds, the **Port Settings** dialog for Multi-ICE then reports that its **Current Port Mode** is standard **8-bit** bidirectional. See *Parallel port settings dialog* on page 3-18.

  — Autodetection of the Multi-ICE interface unit is failing.

  In the **Port Settings** dialog for Multi-ICE, select the specific port that Multi-ICE is using (such as **LPT1**), instead of the default setting of **AUTO**. See *Parallel port settings dialog* on page 3-18.

  — A printer is configured to use the same port as Multi-ICE.

  On the Windows Start menu, choose **Settings** → **Printers**, and open the **Properties** dialog for each printer. If a printer is configured to use the same port as Multi-ICE, reconfigure it to use a different port.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

— There is a conflict or other problem with the port that Multi-ICE is using.

The facilities to determine conflicts and to troubleshoot problems are dependent on the version of Windows that you are using. For more information, refer to the documentation supplied with your computer.

— Another peripheral is interfering with the parallel ports.

IR ports on laptops are particularly likely to do this. Disable these, and then disable any other peripherals that you do not require.

— The port only works correctly with Multi-ICE as a basic unidirectional port.

In the **Port Settings** dialog for Multi-ICE, check the **Force 4-bit access** box. See *Parallel port settings dialog* on page 3-18.

- The server has lost contact with the Multi-ICE interface unit or the target hardware.

  The most common cause is switching on adaptive clocking and failing to provide a good **RTCK** signal. If autoconfiguration is used, the presence of the **RTCK** signal is also autoconfigured. However, if the target circuitry provides an **RTCK** signal during autodetection, but subsequently fails to generate **RTCK**, a hardware interface timeout occurs.

5.1.7 The debugger reports “Unable to set vector catch breakpoints on exception vectors”

On an ARM7-based system, Multi-ICE tries to set software breakpoints on the vector table entries indicated by the value of `vector_catch`. However, if you have ROM at address `0x0`, then Multi-ICE is unable to do this and reports this error message.

On ARM9 family, ARM10 family, and XScale microarchitecture processors, there is built-in vector catch hardware so there is no problem with ROM at zero.

5.1.8 Data aborts or crashes when loading or running applications

The ARM compiler and linker tools use a default code address of `0x8000` so, unless you select an alternative address, there must be usable memory in the target memory map at `0x8000` and above. This is true for the ARM Development Board, the ARM PIE and ARM PIV boards, and the ARM Integrator core modules.

The Multi-ICE DLL has to assume a memory map for the target board when:

- the C library uses semihosting to get stack and heap information
- DCC semihosting is enabled, for the location of the DCC semihosting handler.

By default, the Multi-ICE DLL assumes that there is writable memory between (approximately) `0x60000`, and `0x80000`. These addresses are valid if the target had 512KB RAM located at the bottom of the address map. The DCC semihosting code is downloaded at `0x70000` by default.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To change the location of the stack and heap you must change the debugger variable `top_of_memory`. To change where the DCC semihosting handler is located, you must change the debugger variable `semihosting_dcchandler_address`.

5.1.9 DCC semihosting, the channel viewer, or the DCC fails

The following can be the cause of the problem:

- The processor you are using does not have a DCC, for example the ARM7DI processor that is used on the ARM Evaluation Board (HBI0041).
- You cannot use a channel viewer and DCC semihosting at the same time. Choose one or the other.
- You are using a core with a revision C or earlier AMBA® wrapper.

  September 1998 versions of the Atmel AT91 chip suffer from this problem. There is no known workaround other than to replace the device.

5.1.10 A program that prints strings seems to load and run, but displays garbled text

The target memory controller does not support byte reads. This is a common and serious problem because any C code that accesses **`char`**s does not work correctly.

None of the built-in string handling functions (for example, `strcmp`, `strlen`, and `strcpy`) work properly without byte-readable memory, and accesses to arrays of **`char`**s also fail. It is a misconception that because ARM code is all 32-bit (or all 16-bit for Thumb), that only word and halfword accesses must be supported by the memory controller. Code can be specially written to work with memory controllers designed like this, but processors using these controllers do not run generic C code correctly.

5.1.11 A 'C' program including string handling or uses char arrays works on some ARM processors but not on others

The target memory controller does not support byte reads. This is a common and serious problem because any C code that accesses **`char`**s does not work correctly.

See *A program that prints strings seems to load and run, but displays garbled text* for more details.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.1.12 When trying to connect Multi-ICE and a logic analyzer to an ARM Integrator board to trace a program, Multi-ICE continually reports "The target is being reset, unable to connect"

The pullup fitted to **nSRST** on Integrator CM7TDMI boards is a 47KΩ resistor. This is a relatively weak pullup, and can result in a voltage level on **nSRST** that Multi-ICE detects as a logic zero (a reset condition). This is more likely to occur when a logic analyzer cable is connected to the trace port because the trace port includes the **nSRST** signal.

It might be possible to get Multi-ICE to connect by unplugging the analyzer and plugging it back in again. The permanent solution is to replace the pullup resistor R22 with a 10KΩ device. Follow this procedure to replace the resistor (or contact ARM technical support for assistance):

1. Remove the 47KΩ resistor at position R22. R22 is on the top side of the board near the Multi-ICE connector.
2. Fit a 10KΩ 0603 resistor at position R22.

5.1.13 My application works using ARMulator but quickly crashes when I use Multi-ICE

This fault might be because there is no memory where in memory the application stack has been placed by Multi-ICE, or because stack memory and other memory used by the application overlap.

The default memory model for ARMulator creates memory pages as required through the whole address space, and therefore the ARMulator configuration file can specify a value for top of stack in high memory. Consequently, programs in low memory do not normally interfere with the stack.

Multi-ICE must use whatever memory the target has available, but is never told what the memory map of the target is. Whether this matters depends on the how the application is linked:

Simple semihosted image

The location of the stack is defined by the debugger variable `top_of_memory`. The default value is `0x80000`. It is important that this variable is set correctly.

Scatterloaded The location of the stack and heap is defined by a function called by the C-library, and Multi-ICE is not involved.

Refer to *Internal variable descriptions* on page 4-39 for more information about `top_of_memory` and how the ARM C-library uses it.

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.1.14 Running the Multi-ICE server makes my computer run very slowly

When you run the Multi-ICE server the Multi-ICE parallel port driver is activated. On some machines with more than one parallel port, the parallel port driver can consume a very large proportion of the available processor time in trying to check whether a second parallel port has a Multi-ICE unit connected.

There is no problem if there is only one parallel port.

To avoid this slowdown, explicitly select the parallel port you are using with the Multi-ICE interface unit, as described in *Parallel port settings dialog* on page 3-18. Do not leave the setting on Auto. From Multi-ICE Version 2.1 onwards, this setting is retained between sessions.

5.1.15 I cannot connect to a Multi-ICE server from the computer that is running it

If you start a Multi-ICE server, and later try to connect to it from the same computer, you might get the error message:

```
Initialisation failed: Failed to connect to the server application - check
location of server.
```

If this happens, try to connect from a different computer. If this succeeds, it indicates that the cause of the problem is the type of connection to the server.

Connecting on a local machine uses shared memory transfer by default. The protection for this mechanism is stronger than for RPC access. For example, if you start the server logged in as one user, but then attempt to connect to it as a different user, this fails if you use shared memory transfer, but succeeds if you use RPC.

To fix such problems, when you select the Multi-ICE server to debug (see *Connect configuration tab* on page 4-8), do *not* click the button labeled **This computer**. Instead, you *must* click the **Another computer** button, and then select your workstation in the **Select server location** dialog that appears. This forces RPC access, and so local access to the Multi-ICE server behaves the same way as remote access.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2 Error messages

This section explains the error messages that are displayed in dialog boxes, and provides a way to recover from the error wherever possible. The messages are described in the following sections:

- *Multi-ICE server messages*
- *Multi-ICE DLL messages* on page 5-16.

5.2.1 Multi-ICE server messages

These are the server error messages:

`An error occurred while attempting to set up the TAP configuration.`

The hardware cannot be configured, or the application ran out of memory.

`An error occurred while opening the selected port.`

An unexpected software error has occurred. Contact technical support.

`An error occurred while retrieving the hardware details. This application will now terminate.`

Contact technical support at your supplier.

`An error prevented retrieval of hardware details.`

Contact technical support at your supplier.

`Hardware Interface Timeout.`

See *The debugger reports "Hardware interface timeout"* on page 5-7.

`Auto-Configuration failed. Check the target power is on. Your chip may require manual configuration.`

This might be because:

- The power is off.
- The Multi-ICE directory does not exist, or cannot be written to. Check that the directory exists and that you have sufficient access to it.
- The chip might not be an ARM chip. You must write a manual configuration file. See *Server configuration* on page 3-14.

`Could not clean D-Cache - memory may appear incoherent in writeback regions.`

For targets that have separate data and instruction caches (such as the ARM920T), Multi-ICE uses the **Cache clean code address** setting in the Multi-ICE configuration dialog (see *Processor Settings Tab* on

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

page 4-14) to specify memory it can use to store and execute the *cache cleaning* code. Multi-ICE downloads and runs relocatable code into this memory the first time it is required. If it cannot download the code correctly, it displays this warning message.

Multi-ICE can fail to download the code to the target for one or more of the following reasons:

- There is no memory at the **Cache clean code address**
- The memory at **Cache clean code address** is in read only memory (either true ROM or memory that requires a special write procedure, such as EEPROM)
- There is an active Memory Management Unit (MMU) and the page containing **Cache clean code address** is marked read-only.

You can take one or more of the following steps to correct this:

- You can change **Cache clean code address** to an address in RAM that Multi-ICE can access and that is not used by the application
- You can ensure that the MMU page descriptor for the memory you specified enables data reads, data writes and instruction reads.

You can force Multi-ICE to download the cache clean code again by changing the **Cache clean code address** setting in the Multi-ICE configuration dialog.

If you ignore the error, you must be aware of the following:

- The *Data Cache* (DCache) uncachable bit is not set, and the DCache is not cleaned while in debug state.

  This means that memory is still read precisely as the processor sees it, so there is no inconsistency even when the DCache has dirty data in it. However, any data read in cachable regions causes DCache linefills to occur, so new addresses are written into the DCache and old data is evicted.
- Any data that is written to memory is written as normal.

  This means that any writes that hit addresses in the DCache do not get into memory, but only into the cache. This matters if the data is actually code.

  A side-effect of this is that software breakpoints set on addresses that have been cached in the DCache are not read correctly on processor instruction fetches and so are not taken.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

If the *Fault Address Register* (FAR) and *Fault Status Register* (FSR) are implemented by the system coprocessor, they are read on debug state entry. The values are available in the system coprocessor (number 15) register display and get restored before leaving debug state, unless you write new values. If you perform a debug action that causes a memory abort, then an error is returned that includes the FSR and FAR values just after the abort.

`TAP configuration failed. Please check that the target hardware and the Multi-ICE unit are properly connected and powered up.`

Check that the Multi-ICE interface unit has power, and is connected to the workstation using the cable supplied with Multi-ICE.

The Multi-ICE parallel port driver requires that the parallel port interface on the host workstation is at one of the two standard addresses. This is `0x378` for LPT1, and `0x278` for LPT2. If your hardware uses a different address, the Multi-ICE interface unit cannot be used.

If you are using Multi-ICE with a 14-way connector on the target board:

- If power is being supplied by the target, check that jumper J3 on the 14-way JTAG adaptor is linking **+5V** and **TP** and that, if necessary, the series resistor on your target has been shorted. See *Connecting to nonstandard hardware* on page 2-11.
- If power is being supplied externally, check that the supply voltage is correct and switched on. See *Connecting the Multi-ICE hardware* on page 2-6.

`Failed to open parallel port. The port may be in use.`

This might be because:

- another device is using the parallel port (for example, a printer)
- you have faulty parallel port hardware
- there are other problems or incompatibilities with the parallel port on your PC, as described in *The debugger reports "Hardware interface timeout"* on page 5-7.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`Multi-ICE device driver failure. Please check that the Multi-ICE device driver is installed and running and there is no other Multi-ICE Server running.`

The Multi-ICE parallel port driver is not present or has not started. The driver is started in different ways depending on the operating system:

Windows NT 4.0 In the devices control panel look in the device list for the name `Multi-ICE` and check the driver is present and has started.

Windows 95, 98, Me, and 2000

The Multi-ICE driver is started automatically by the server.

`The server could not initialize correctly. Please close down one or more applications and re-start.`

Other applications are using system resources required by the Multi-ICE server, for example, system timers, graphics resources, or main memory. Close down other applications if there are any. If not, the error might indicate that Windows must be restarted.

`The server installation is incomplete or damaged. You may wish to re-install the software.`

The server cannot find a required entry in the system registry. This might happen if:

- this version of Multi-ICE has not been installed on this computer
- the Windows system registry of the computer has been damaged.

If you have an existing installation of Multi-ICE, you must first uninstall it. You must then reinstall the Multi-ICE software.

Note

It is recommended that you check for any wanted configuration files stored in the Multi-ICE directory before uninstalling.

`The attached device is not compatible with this server.`

The hardware attached to the parallel port is either:

- not a Multi-ICE interface unit
- a faulty Multi-ICE interface unit.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Check that the Multi-ICE interface unit is connected to the correct parallel port. If it is, check that the unit is connected as indicated in *Connecting the Multi-ICE hardware* on page 2-6 and that the cable is working properly. If the error persists you must contact technical support at your supplier.

`The decision to read core information has been re-specified in file filename at line number.`

It has been specified more than once in the configuration file whether core information is automatically read from the connected devices. You must remove one of the two entries from the configuration file.

`The selected server is not compatible with this debugger.`

The ARM Multi-ICE DLL can only be used with the ARM Multi-ICE server.

5.2.2 Multi-ICE DLL messages

These are the debugger error messages:

`The driver for device could not be found.`

The `.MUL` file corresponding to the device does not exist in the expected location.

`The driver for device could not be found. This was due to an installation problem. Please re-install the Multi-ICE software.`

The Multi-ICE DLL cannot find an entry in the registry.

`The driver for device could not be read.`

This is because:

- the `.MUL` file corresponding to the device has been opened by another application
- the `.MUL` file does not exist
- the `.MUL` file is corrupted or out of date.

`The format of the driver for device is not valid.`

The `.MUL` file corresponding to the device has been corrupted, or the `.MUL` file for the device is out-of-date. The installation has probably become corrupted. Reinstall the software.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`The previously connected device is no longer available.`

This is because the server that you were connected to when the debugger session was last saved is no longer connected to the same processor.

`The server found on location has not been initialized.`

The server was found, but it had not been configured (either by automatic or manual configuration). The server window looks similar to Figure 2-6 on page 2-16. Configure the server and try connecting again.

`The server on location returned an unexpected error of [number]. The server may be incompatible.`

The Multi-ICE DLL and Multi-ICE server versions are incompatible. Display the **Help → About Multi-ICE** dialog on the Multi-ICE server to determine the version you are using and compare it to the version number in the title string of the Multi-ICE DLL configuration window. Upgrade or reinstall the software as required to bring the server in line with the DLL.

`The server on location returned too much driver information. The version of the server software may not be compatible with your current debugger.`

The Multi-ICE DLL and Server versions are incompatible. You must use the Multi-ICE server **Help → About Multi-ICE** dialog to determine the server version you are using and upgrade or reinstall the software.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 6
System Design Guidelines

This chapter provides information on developing ARM-based devices and PCBs that can be debugged using Multi-ICE. It contains the following sections:

- *About the system design guidelines* on page 6-2
- *System design* on page 6-3
- *ASIC guidelines* on page 6-9
- *PCB guidelines* on page 6-12
- *JTAG signal integrity and maximum cable lengths* on page 6-15
- *Compatibility with EmbeddedICE interface target connectors* on page 6-17.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.1 About the system design guidelines

This chapter describes the following:

- How to connect multiple TAP controllers in systems comprising more than one unit. For example, connecting an ARM core plus a *Digital Signal Processor* (DSP).
- Support for demand-paged systems.
- Using the Multi-ICE adaptive clocking feature to control the JTAG clock rate.
- Reset signals, providing examples of circuits.
- Compatibility with EmbeddedICE connectors.

Refer to Appendix F *JTAG Interface Connections* for details of the JTAG interface connector pinout.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.2 System design

This section describes how to design clocking and reset circuits that are compatible with Multi-ICE. It contains the following sections:

- *Mixing ARM cores with other devices*
- *Using adaptive clocking to synchronize the JTAG port*
- *Reset signals* on page 6-6.

6.2.1 Mixing ARM cores with other devices

You can use Multi-ICE to debug systems that mix ARM processor cores with other devices. The TAP controllers must be daisy-chained as described in *ICs containing multiple devices* on page 6-9. When accessing a particular device, Multi-ICE places all other TAP Controllers in bypass mode. Multi-ICE is shipped to you with the instruction register lengths for many ARM TAP controllers in a database file `IRlength.arm`. You must add entries to the database for any TAP controllers you use that are not present in this database.

Multi-ICE provides a software interface layer that enables you to write drivers to access non-ARM devices. For more information, refer to the *Multi-ICE TAPOp API Reference Guide*.

6.2.2 Using adaptive clocking to synchronize the JTAG port

ARM-based devices using only hard macrocells, for example ARM7TDMI and ARM920T, use the standard five-wire JTAG interface (**TCK**, **TMS**, **TDI**, **TDO**, and **nTRST**). Some target systems, however, require that JTAG events are synchronized to a clock in the system. To handle this case an extra signal is included on the JTAG port. For example, this synchronization is required in:

- an ASIC with single rising-edge D-type design rules, such as one based on an ARM7TDMI-S processor core
- a system where scan chains external to the ARM macrocell must meet single rising-edge D-type design rules.

The adaptive clocking feature of Multi-ICE addresses this requirement. When adaptive clocking is enabled, Multi-ICE issues a **TCK** signal and waits for the **RTCK** (Returned **TCK**) signal to come back. Multi-ICE does not progress to the next **TCK** until **RTCK** is received.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

- If you use the adaptive clocking feature, transmission delays, gate delays, and synchronization requirements result in a lower maximum clock frequency than with non-adaptive clocking. Do not use adaptive clocking unless it is required by the hardware design.
- If, when autoconfiguring a target, the Multi-ICE interface unit receives pulses on **RTCK** in response to **TCK** it assumes that adaptive clocking is required, and enables adaptive clocking in the target configuration. If the hardware does not require adaptive clocking, the target is driven slower than it could be. You can disable adaptive clocking using controls on the JTAG settings dialog.

You can use adaptive clocking as an interface to targets with slow or widely varying clock frequency, such as battery-powered equipment that varies its clock speed according to processing demand. In this system, **TCK** might be hundreds of times faster than the system clock, and the debugger loses synchronization with the target system. Adaptive clocking ensures that the JTAG port speed automatically adapts to slow system speed.

Figure 6-1 illustrates a circuit for basic applications, with a partial timing diagram shown in Figure 6-2 on page 6-5. The delay can be reduced by clocking the flip-flops from opposite edges of the system clock, because the second flip-flop only provides better immunity to metastability problems. Even a single flip-flop synchronizer never completely misses **TCK** events, because **RTCK** is part of a feedback loop controlling **TCK**.

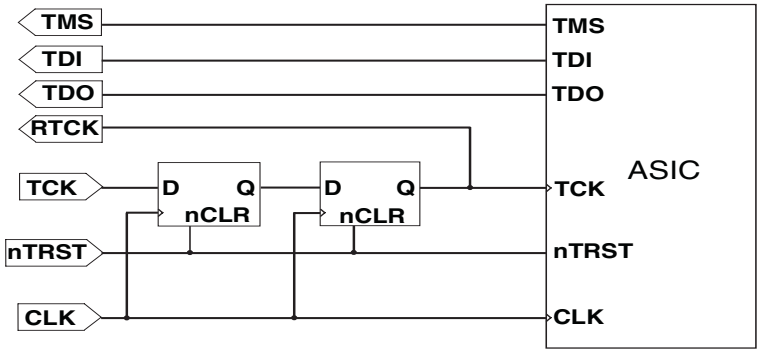


Figure 6-1 Basic JTAG port synchronizer

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

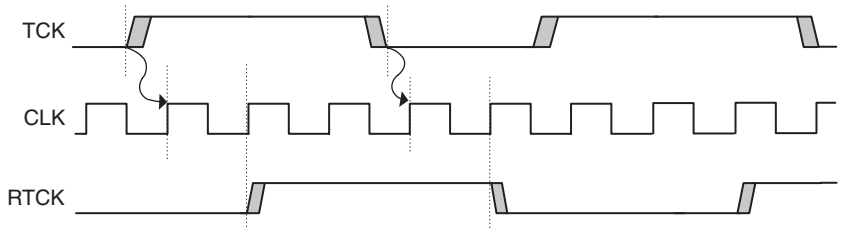


Figure 6-2 Timing diagram for the Basic JTAG synchronizer in Figure 6-1 on page 6-4

It is common for an ASIC design flow and its design rules to impose a restriction that all flip-flops in a design are clocked by one edge of a single clock. To interface this to a JTAG port that is completely asynchronous to the system, it is necessary to convert the JTAG **TCK** events into clock enables for this single clock, and to ensure that the JTAG port cannot overrun this synchronization delay. Figure 6-3 shows one possible implementation of this circuit, and Figure 6-4 on page 6-6 shows a partial timing diagram, showing how **TCKFallingEn** and **TCKRisingEn** are each active for exactly one period of **CLK**. It also shows how these enable signals gate the **RTCK** and **TDO** signals so that they only change state at the edges of **TCK**.

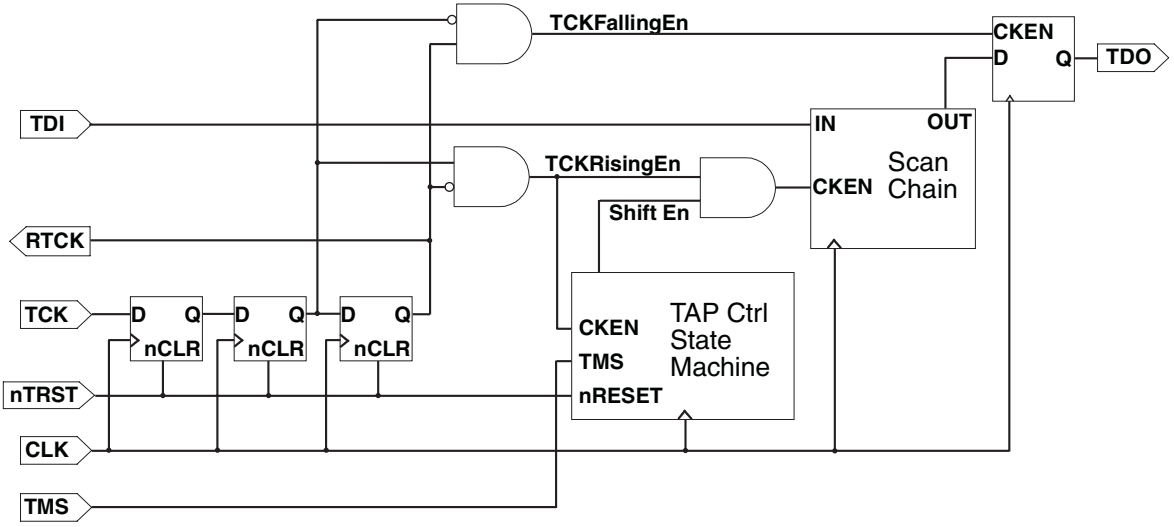


Figure 6-3 JTAG port synchronizer for single rising-edge D-type ASIC design rules

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

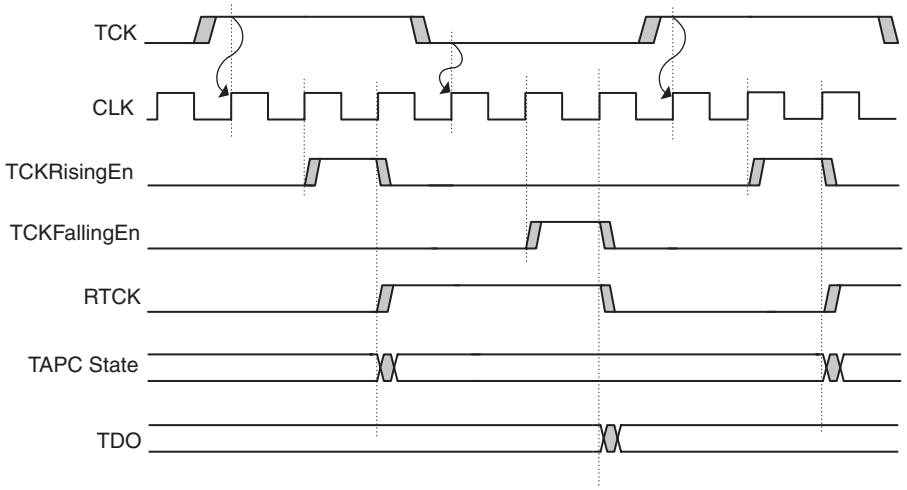


Figure 6-4 Timing diagram for the D-type JTAG synchronizer in Figure 6-3 on page 6-5

6.2.3 Reset signals

This section describes the reset signals that are available on ARM devices and how Multi-ICE expects them to be wired. It is presented in the following sections:

- *ARM reset signals*
- *Multi-ICE reset signals* on page 6-7
- *Example reset circuits* on page 6-7.

ARM reset signals

All ARM cores have a main processor reset that might be called **nRESET**, **BnRES**, or **HRESET**. This is asserted by one or more of these conditions:

- power on
- manual push button
- remote reset from the debugger (using Multi-ICE)
- watchdog circuit (if appropriate to the application).

Any ARM processor core including the JTAG interface has a second reset input called **nTRST** (TAP Reset). This resets the EmbeddedICE logic, the TAP controller, and the boundary scan cells. This is activated by one or more of these conditions:

- power on
- remote JTAG reset (from Multi-ICE).

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

It is strongly recommended that both signals are separately available on the JTAG connector. If the **nRESET** and **nTRST** signals are linked together, resetting the system also resets the TAP controller. This means that:

- it is not possible to debug a system from reset, because any breakpoints previously set are lost
- you might have to start the debug session from the beginning, because Multi-ICE does not recover when the TAP controller state is changed.

Multi-ICE reset signals

The Multi-ICE interface unit has two reset signals connected to the debug target board:

- **nTRST** drives the JTAG **nTRST** signal on the ARM processor core. It is an open collector output that is activated whenever the Multi-ICE software has to re-initialize the debug interface in the target system.
- **nSRST** is a bidirectional signal that both drives and senses the system reset signal on the target board. The open collector output is driven LOW by the debugger to re-initialize the target system.

The target board must include a pull-up resistor on both reset signals.

Example reset circuits

The circuits shown in Figure 6-5 on page 6-8 and Figure 6-6 on page 6-8 illustrate how the behavior described in *Reset signals* on page 6-6 can be achieved. The MAX823 used in Figure 6-6 on page 6-8 is a typical power supply supervisor. It has a current limited **nRESET** output that can be overdriven by the Multi-ICE interface unit.

When the Multi-ICE server detects a reset, it records this event so that clients can find out about it when they next ask for the target status. The record is kept for each active connection, so that one client cannot prevent another client from finding out about the reset. This is particularly useful if the target system is using a watchdog reset circuit because there might be no other evidence that the system has reset.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

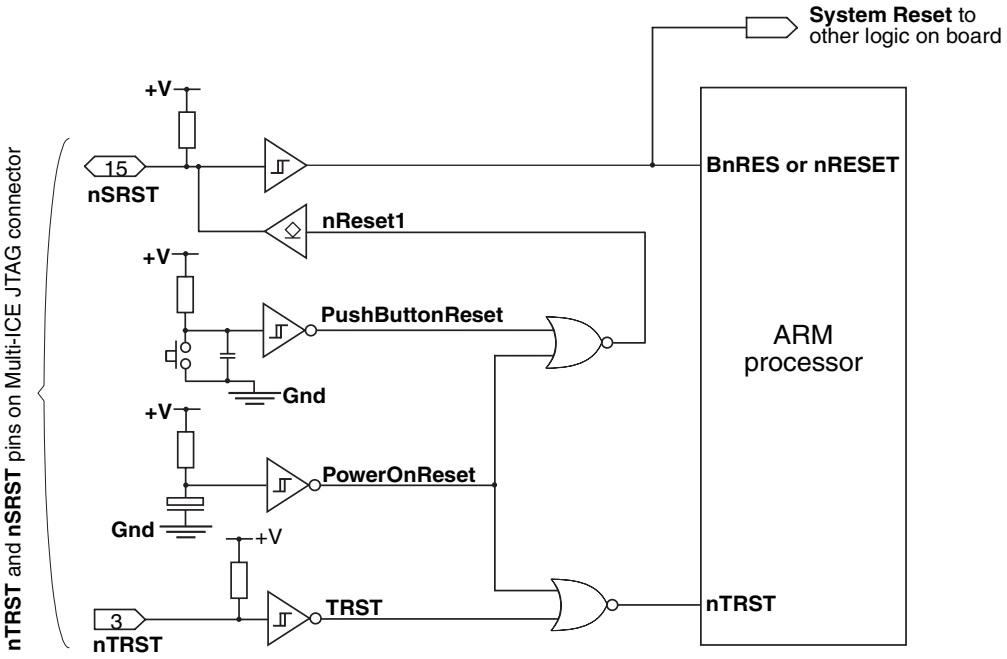


Figure 6-5 Example reset circuit logic

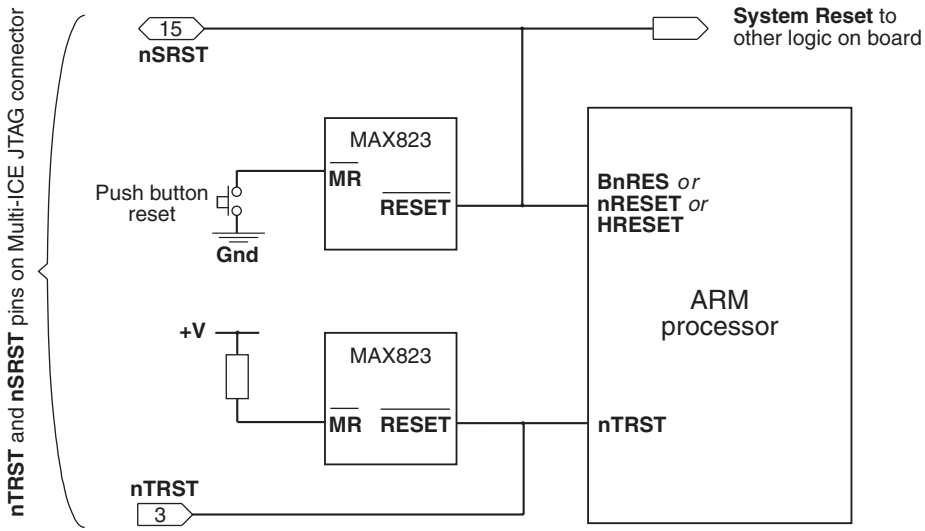


Figure 6-6 Example reset circuit using power supply monitor ICs

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.3 ASIC guidelines

This section describes:

- *ICs containing multiple devices*
- *Constraints imposed by the Multi-ICE server* on page 6-10
- *Boundary scan test vectors* on page 6-11.

6.3.1 ICs containing multiple devices

The JTAG standard originally described daisy-chaining multiple devices on a PCB. This concept is now extended to multiple cores within a single package. If more than one JTAG TAP controller is present within your ASIC, they must all be serially chained so that Multi-ICE can communicate with all of them simultaneously. The chaining can either be within the ASIC or external to it.

There are a few possible configurations of multiple TAP controllers:

- TAP controllers serially-chained within the ASIC
- each set of JTAG connections is pinned out separately
- multiplexing of data signals.

TAP controllers serially chained within the ASIC

This is the natural extension of the JTAG board-level interconnection, and is the one recommended for use with Multi-ICE. There is no increase in package pin count, and only a very small impact on speed because unaddressed TAP controllers can be put into bypass mode, as shown in Figure 6-7 on page 6-10.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

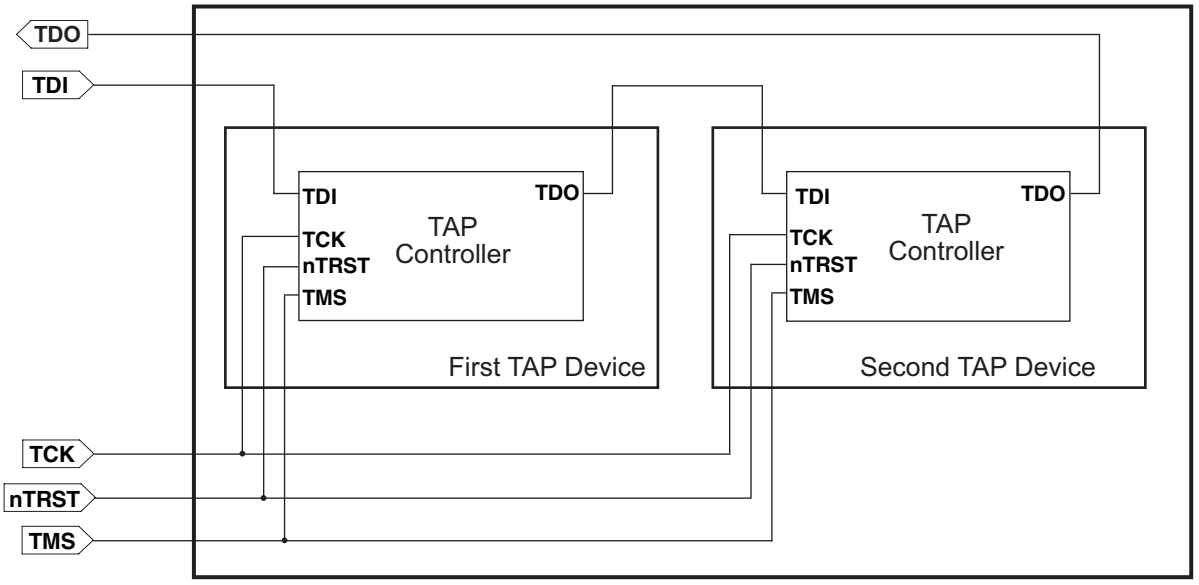


Figure 6-7 TAP Controllers serially chained in an ASIC

Each set of JTAG connections is pinned out separately

This gives the greatest flexibility on the PCB, but at the cost of many pins on the device package. If this method is chosen to simplify device testing, the JTAG ports must be serially chained on the PCB when Multi-ICE is to be used. The separate JTAG ports can be tracked to separate headers on the PCB, but this then requires one Multi-ICE interface unit per header, and is unnecessary.

Multiplexing of data signals

There is no support in Multi-ICE for multiplexing **TCK**, **TMS**, **TDI**, **TDO**, and **RTCK**, between a number of different processors.

6.3.2 Constraints imposed by the Multi-ICE server

The Multi-ICE Server has a number of design constraints that may affect the ability to debug your system.

The following constraints apply to all systems:

- The IR (*Instruction Register*) for each TAP controller must be between 2 and 63 bits long (inclusive).

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- The maximum total length of the IR scan chain is 64 bits.
- The maximum number of TAP controllers is 64.

The following constraints apply only if your TAP controller contains an SCSR (*Scan Chain Select Register*):

- The length of the SCSR for each TAP controller must not exceed either of the following:
  - 32 bits
  - (65 - (*number of TAP controllers*)) bits
- The maximum scan chain number is 63.

6.3.3 Boundary scan test vectors

If you use the JTAG boundary scan test methodology to apply production test vectors, you might want to have independent external access to each TAP controller. This avoids the requirement to merge test vectors for more than one block in the device. One solution to this is to adopt a hybrid, using a pin on the package that switches elements of the device into a test mode. This can be used to break the internal daisy chaining of **TDO** and **TDI** signals, and to multiplex out independent JTAG ports on pins that are used for another purpose during normal operation.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.4 PCB guidelines

This section contains guidelines on the physical and electrical connections present on the target PCB:

- *PCB connections*
- *Target interface logic levels*.

For Multi-ICE JTAG header connectors refer to *Multi-ICE JTAG interface connections* on page F-2.

6.4.1 PCB connections

It is recommended that you place the JTAG header as closely as possible to the target device, as this minimizes any possible signal degradation due to long PCB tracks.

Figure 6-8 shows the layout of possible PCB connections.

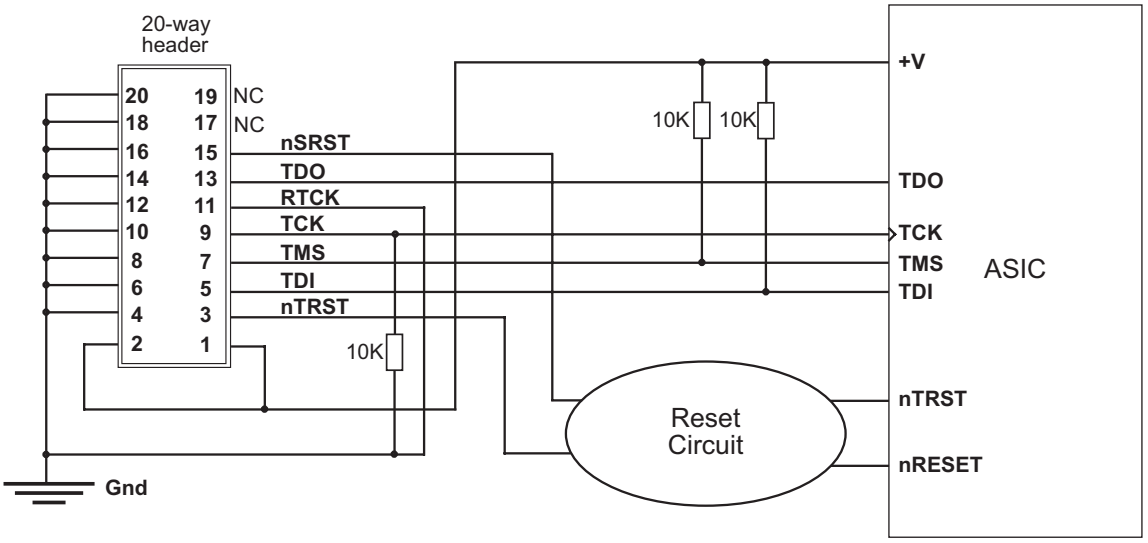


Figure 6-8 Typical PCB connections

6.4.2 Target interface logic levels

Multi-ICE is designed to interface with a wide range of target system logic levels. It does this by adapting its output drive and input threshold to a reference voltage supplied by the target system.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VTref (pin 1 on the JTAG header connector) feeds the reference voltage to the Multi-ICE interface unit. This voltage, clipped at approximately 3.2V, is used as the output high voltage (**Voh**) for logic 1s (ones) on **TCK**, **TDI**, and **TMS**. 0V is used as the output low voltage for logic 0s (zeroes). The input logic threshold voltage (**Vi(th)**) for the **TDO**, **RTCK**, and **nSRST** inputs is 50% of the **Voh** level, and so is clipped to approximately 1.55V. The relationships of **Voh** and **Vi(th)** to **VTref** are shown in Figure 6-9.

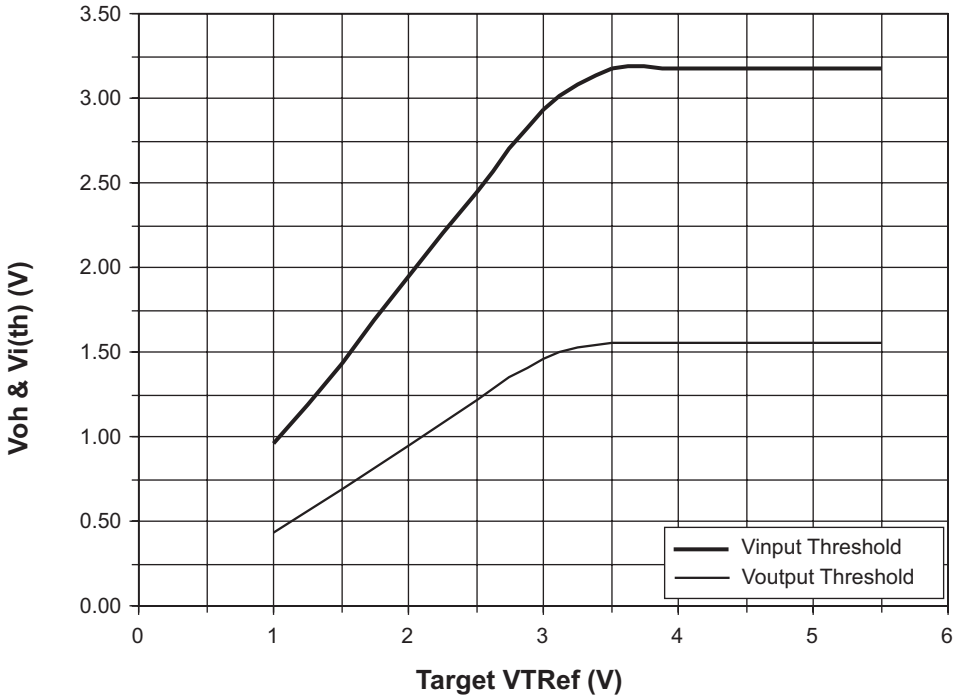


Figure 6-9 Target interface voltage levels

The adaptive interface levels work down to **VTref** less than 1V. If, however, **VTref** becomes less than approximately 0.85V, Multi-ICE interprets this condition as Target Not Present, and the software reports this as an error condition.

The **nTRST** output from Multi-ICE is effectively driven *open collector*, so it is actively pulled to 0V but relies on a pull-up resistor within the target system to end the reset state. This is because it is common to wire-OR this signal with another source of **nTRST**, such as power-on reset in the target system.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The **nSRST** output from Multi-ICE is similarly driven open collector, and must be pulled-up with a resistor in the target system. As this signal is also an input to the Multi-ICE interface unit, there is a large-value internal pull-up resistor (51kΩ to **Voh**). This is to avoid spurious lows on the input when **nSRST** is not connected to the target system.

The input and output characteristics of the Multi-ICE interface unit are compatible with logic levels from TTL-compatible, or CMOS logic in target systems. For information when assessing compatibility with other logic systems, the output impedance of all signals is approximately 100Ω.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.5 JTAG signal integrity and maximum cable lengths

For JTAG-based debugging, you must have a very reliable connection between Multi-ICE and the target board because there is no way to detect or correct errors. For this reason it is important to guarantee good signal integrity.

One factor that can limit the maximum cable length is propagation delays. Normally the Multi-ICE interface unit samples data returning from the target using the same clock as for sending data, **TCK**. If the propagation delay gets too long then the Multi-ICE interface unit samples the signal at the wrong time. This can be resolved by using *adaptive clocking*. In this mode the target returns a clock, **RTCK**, and Multi-ICE does not sample data on **TDO**, or send further data on **TDI**, until clocked by this signal.

In an ASIC or ASSP (for example, in ARM processor based microcontrollers) the **TDO** and **RTCK** signals are not typically implemented with a stronger driver than other signals on the device. The strength of these drivers varies from device to device. An example specification is to sink or source 4mA. Many designs connect these pins on the device directly to the corresponding pins on the Multi-ICE connector.

Over very short lengths of cable, such as the one supplied with Multi-ICE, this type of weak driver is adequate. However, if longer cables are used then the cable becomes harder to drive as the capacitive load increases. When using longer cables it becomes essential to consider the cable as a transmission line and to provide appropriate impedance matching, otherwise reflections occur.

Multi-ICE has much stronger drivers and they are connected through 100Ω series resistors to impedance match with the JTAG cable. The output circuitry of Multi-ICE can easily sink or source over 40mA of current. This is very much better than the typical circuit used at the target end.

With the typical situation at the target end (weak drivers, no impedance matching resistors) you can only expect reliable operation over short cables (approximately 20cm). If operation over longer cables is required you must improve the circuitry used at the target end.

The recommended solution is to add an external buffer with good current drive and a 100Ω series resistor for the **TDO** (and **RTCK** if used) signals on your target board. Using this technique you can debug over a significantly longer cable, up to several metres. Depending on cable length and propagation delays through your buffers and cables it might still be necessary to use adaptive clocking.

If you are not already using adaptive clocking in your design, you can generate **RTCK** at the target end by using the **TCK** signal fed through the same buffer and impedance matching circuit as used for **TDO**. If even longer cables are required, another solution is to buffer the JTAG signals through differential drivers, for example, RS422 and

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

connect to differential receivers at the remote end using twisted pair cable. You must use adaptive clocking to allow for propagation delays in the cable and drivers. Reliable operation is possible over tens of metres using this technique.

Reducing the JTAG clock speed in the Multi-ICE server avoids some, but not all, of the problems associated with long cables. If reducing the speed of downloading code and reading memory in the debugger is not a significant problem, try experimenting with lowering this clock speed.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.6 Compatibility with EmbeddedICE interface target connectors

The EmbeddedICE Interface Unit uses a 14-way connector for the interface to the target system. ARM Limited provides an adaptor board, HPI-0027, so that the Multi-ICE interface unit can be connected to target boards with 14-way connectors. On request, ARM can also supply a board that allows you to connect EmbeddedICE Interface Units to targets using the 20-way connector.

6.6.1 Adaptor to connect a Multi-ICE interface unit to 14-way connectors

The 14-way socket on the adaptor board plugs into a target board with the older header, and the Multi-ICE ribbon cable is connected to the 20-way header on the adaptor board.

The three-pin header J3 has the following connections:

Pin 1 0V, **Gnd**.

Pin 2 Multi-ICE connector pin 2, **Vsupply**.

Pin 3 Target connector pin 1, **SPU**.

The jumper link supplied on the adaptor board connects pins 2 and 3 so that the Multi-ICE interface unit draws its power from the target system in the normal manner. If the target system cannot source a suitable voltage or current, an external 2V to 5V DC supply can be connected to pins 1 and 2.

The three-pin header J4 has the following connections:

Pin 1 0V.

Pin 2 Multi-ICE connector pin 11, **RTCK**.

Pin 3 Resistor fed by Multi-ICE connector pin 9, **TCK**.

The jumper link supplied on the adaptor board connects 0V back to the Multi-ICE **RTCK** input. If the target system is to use adaptive clocking, **TCK** can be tapped off here, and the synchronized version used to clock the target can be fed back as **RTCK**.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix A
Server Configuration File Syntax

This appendix documents the server configuration file format. It includes the following sections:

- *IR length configuration file* on page A-2
- *Device configuration file* on page A-3.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.1 IR length configuration file

Multi-ICE must calculate the length of the scan chain it is connected to. It does this by adding the length of each TAP IR register it finds in the scan chain. This is possible by identifying the devices in the chain and using information in a file called `IRlength.arm`. This file is stored in the Multi-ICE installation directory. An extract from the `IRlength.arm` file is shown in Example A-1.

Example A-1 Extract from IRlength.arm configuration file

```
;ARM7 series cores
ARM7TDMI=4
ARM7TDMI-S=4
ARM740T=4
;ARM9 series cores
ARM9TDMI=4
ARM920T=4
```

A.1.1 File syntax

The file syntax is:

- Blank lines are ignored.
- Comments begin with a semicolon and continue to the end of the line.
- Other text in the file must have the form `name=value`, where `name` is the name of a device, and `value` is the length of the IR for that device. Spaces are not allowed between `name` and `value`.

The file provided with the product includes all the devices that are recognized by the Multi-ICE DLL.

A.1.2 Device aliases

You can add an entry to the `IRlength.arm` file with an IR length value of 0. You can then use the name in a manually-produced configuration file as an alias for another device. This is useful, for example, when two cooperating clients require access to the same device. By using an alias the server allows both clients to connect. Using the `deselect` parameter in TAPOp calls enables the two clients to arbitrate access to the device.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 Device configuration file

A device configuration file is a text file containing the information required to set up the Multi-ICE server for a particular target. You can store configuration files wherever it is convenient.

A.2.1 Syntax

A device configuration file is a text file with the file suffix `.cfg`. The syntax is similar to the Windows INI file format:

- Blank lines are ignored.
- Comments begin with a semicolon and continue to the end of the line.
- Text enclosed in square brackets `[]` indicates the start of a section of the file and must be one of the following strings (case is not important):
  - `TITLE`
  - `TAP` *`n`*
  - `RESET`
  - `TIMING`
  - `TAPINFO`

  The *`n`* in `Tap` *`n`* must be the number of the TAP controller you are describing, starting at 0.

The contents of each section of the configuration file is as follows:

Title

The title string.

Tap *n*

The name of the device (which must have an entry in `IRlength.arm`) as shown in Example A-2 on page A-4. If the software requires a device alias this must be placed on the next line in the same format. The syntax is:

`Name DriverOptions``, `*`DriverVersion`*

The *`DriverOptions`* field is an optional text string passed to the driver for a specific device. There must be a space between the driver name and the driver options, but the driver options can contain spaces. The field is terminated by a comma or newline.

If there is at least one asterisk `*` in the option string, the Multi-ICE server does not display the status indicator `[X]` in the TAP window.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The *DriverVersion* field is an integer number. Specifying a value indicates that only drivers with this version number or higher are acceptable.

Example A-2 Configuration file TAP section

```
[TAP 0]
ARM7TDMI        ;Plain, no options
[TAP 1]
ARM7TDMI , 2    ;Requires V2 or higher
[TAP 2]
ARM7TDMI *ABC   ;Driver=ARM7TDMI, Options="ABC", no status
```

Reset

The section contains one or both of the words, as shown in Example A-3:

nSRST Resetting the system involves asserting **nSRST**.

nTRST Resetting the system involves asserting **nTRST**.

If no reset section is present then resetting the system using the Multi-ICE server GUI asserts both signals.

Example A-3 Configuration file Reset section

```
[Reset]
nTRST
nSRST
```

Timing

The timing section defines how the **TCK** signal is generated as shown in Example A-4 on page A-5. The following name value pairs are allowed in this section:

High The period for which **TCK** is HIGH (positive voltage), specified as a value from *TCK frequencies* on page F-7.

Low The period for which **TCK** is LOW (zero voltage), specified as a value from *TCK frequencies* on page F-7.

Adaptive `YES` if the target drives **RTCK**, `NO` if it is not driven.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The HIGH and LOW values correspond to those in the server **Settings → JTAG Settings** dialog box.

Example A-4 Configuration file Timing section

```
[TIMING]
High=100
Low=50
Adaptive=ON
```

Tapinfo

The TAPINFO section (see Example A-5) tells the server whether to read additional information from the device after loading the configuration file. If the section contains YES then the server reads any additional information, otherwise it does not. You can view the information by double-clicking on a TAP controller within the server Configuration window.

Example A-5 Configuration file Tapinfo section

```
[TAPINFO]
YES
```

A TAPINFO option is included in a configuration file to provide flexibility for ASIC developers during testing. When the **Auto-configure** facility is used, TAPINFO is always provided. By default, when loading a configuration file TAPINFO is not provided.

A.2.2 Example configuration file

Example A-6 shows a configuration file with all types of data.

Example A-6 A complete configuration File

```
[TITLE]
Example TAP Configuration
[TAP 0]
ARM7TDMI
[TAP 1]
ARM9TDMI
[Reset]
                ; When you reset:
```


Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
nTRST           ;   reset tap controller
nSRST           ;   and reset the target board
[TIMING]
High=100        ; These HIGH and LOW values correspond to
Low=50          ; those in the server Settings->JTAG Settings
                ; dialog box.
Adaptive=ON     ; Use the RTCK adaptive clocking signal.
[TAPINFO]
YES             ; Tells the server to gather core information
                ; after loading this.cfg file.
```


Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix B
Breakpoint Selection Algorithm

This appendix contains a description of the breakpoint allocation algorithm. It might help you to make the most of the limited breakpoint resources present in a processor. It contains the following sections:

- *Multi-ICE internal breakpoints* on page B-2
- *How the debugger steps and runs code* on page B-4.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

B.1 Multi-ICE internal breakpoints

Multi-ICE maintains a list of the currently requested breakpoints, marking each of them as being internal or external. External breakpoints and watchpoints (those that are explicitly set by you, or the `main` breakpoint that is set on your behalf) are displayed in the debugger lists of breakpoints and watchpoints. Internally set breakpoints are not displayed by the debugger. Internal breakpoints do, however, potentially compete for the breakpoint resources available on the processor, and so can affect you. The use of these internal breakpoints is described below:

Vector catch breakpoints

On an ARM7T-based core vector catch is implemented using breakpoints. This means that breakpoints you set and vector catch breakpoints compete for the hardware breakpoint resources. With the default `vector_catch` setting and hardware, most or all of the vector catch breakpoints are set as soft breakpoints because there are only two breakpoint units available.

One effect of this is that a system with ROM at address 0 runs out of hardware breakpoint units for vector catch, resulting in the following error message:

```
Unable to set breakpoints on exception vectors as specified by
vector_catch
```

To avoid this, run with RAM at address 0 or reduce the number of breakpoints required by changing `vector_catch`.

On XScale microarchitecture processors, ARM9 processor cores, and ARM10 processors, vector catch is implemented in special hardware in the core. Therefore there is no competition between vector catch and your breakpoints, so the above restriction does not apply.

Vector catch breakpoints are set when the debugger starts up and when `vector_catch` is changed.

Semihosting SWI breakpoint

The breakpoint on the SWI vector is treated as a special case. Its allocation is controlled by two debugger internal variables:

- `vector_catch`
- `semihosting_enabled`.

By default the `S` bit in `vector_catch` is not set and `semihosting_enabled` is `1` which means there is a breakpoint on the SWI vector but it is treated as a semihosting request when hit, rather than just stopping.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the S bit in vector_catch is set then Multi-ICE takes a breakpoint on SWI instructions and ignores the setting of semihosting_enabled. If the S bit in vector_catch is unset, the setting of semihosting_enabled determines how Multi-ICE responds to subsequent SWI instructions.

Stepping breakpoints

When stepping through code the debugger sets a breakpoint where the processor must stop. This is also done when the execution type **Run to cursor** is requested.

Single-stepping breakpoints are not set until you request the processor to starts execution. This means that the exact allocation of your breakpoints to hard or soft breakpoints shown in the breakpoint views of the debugger changes when stepping code.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

B.2 How the debugger steps and runs code

The following algorithm is used to step code:

1. If the current instruction has a breakpoint on it, it is removed. If it was a breakpoint you defined, it is put back after the step. If it was a breakpoint from a previous step, it is discarded.
2. The instruction is read and decoded (the decoding scheme chosen is based upon the current value of the T bit in the CPSR). The address of the next instruction to be executed is calculated and a breakpoint placed on that address. Typically this is pc+4 for ARM code and pc+2 for Thumb® code but if a branch instruction is to be stepped the branch address is calculated.
3. The processor is restarted. If an interrupt is pending at this point and interrupts are enabled, the interrupt is taken, executed to completion and the single instruction is stepped after the ISR has completed. Consequently, if an ISR fails to complete, the step also fails to complete.
4. Single stepping a branch to itself is not supported by this algorithm so an error is generated in this case.

The following algorithm is used to run code:

1. If the current instruction has no breakpoint on it the processor is restarted.
2. If the current instruction has a stepping breakpoint on it, it is removed and the processor is restarted.
3. If the current instruction has your breakpoint on it, it is removed and a single step is performed to the next location using the stepping algorithm above. Your breakpoint is put back and the processor is restarted from the new location (unless there is also one of your breakpoints on the new location).

   This is done so that restarting from a breakpoint does not miss any subsequent hits on the same breakpoint. For example, when in a loop, you might execute one pass of the loop by pressing **Go**. This mechanism ensures that only one loop is executed each time you press **Go**.

Stepping through high-level language code involves a combination of these techniques.

Note

Multi-ICE does not rewrite software breakpoint instructions every time you press **Go** in the debugger. This means that if you are using scatter-loaded images or self-modifying code you must manually set and unset software breakpoints around code that changes.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

B.3 Breakpoint and watchpoint allocation algorithm

This simplified algorithm illustrates the basic process that Multi-ICE follows to map your breakpoints and debugger internal breakpoints on to the available hardware resources:

1. Any hardware breakpoint resources currently allocated to you (see `icebreaker_lockedpoints`) are not used. They are considered as unavailable to Multi-ICE and are removed from the list of available hardware.
2. Any watchpoints that have been requested are allocated next, followed by any breakpoints which are on ROM. (Both of these types require dedicated hardware resources.) If there are not enough hardware breakpoint units to set all of these then an error indicating that no more breakpoints or watchpoints can be set is displayed.
3. The remaining breakpoints are assigned as hardware or software breakpoints. Preference is given to your breakpoints over debugger internal breakpoints, and your preference is taken from the value of `sw_breakpoints_preferred`, as follows:
   - If it is zero (the default), your breakpoints are allocated as hardware breakpoints if any breakpoint units are still free, or software if not.
   - If it is nonzero, your breakpoints are allocated as software breakpoints if possible. If it is not possible (for example, the location is in ROM), but a hardware breakpoint can be used instead, then it is used.

   If it is not possible to set a breakpoint, an error indicating that no more breakpoints or watchpoints can be set is displayed.
4. Stepping breakpoints are allocated as hardware breakpoints if possible. This is so that:
   - stepping in/out of ROM does not change the allocation of every breakpoint
   - performance is improved.

For example, with a standard ARM7 core, assuming that none of your breakpoints are already set, `vector_catch`=0 and `semihosting_enabled`=0 (so no internal breakpoints are set), the behavior as breakpoints are added is as follows:

1. The first of your breakpoints is hardware.
2. When a second breakpoint is set, the first breakpoint is downgraded to software (if possible) and the second is made hardware.
3. When subsequent breakpoints are set, the last breakpoint is hardware and all others are software.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix C
Command-line Syntax

This appendix describes the command-line syntax for the Multi-ICE server:

- *Multi-ICE server* on page C-2.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C.1 Multi-ICE server

You can start the Multi-ICE server from the DOS command line (from the Win9x `MS-DOS Prompt`, the Windows NT, or Windows 2000 `Command Prompt`) or by using **Start → Run** to issue an individual command. It is a full Win32 API program.

It understands the following syntax:

```
Multi-ICEServer [-A | config_file]
```

where:

`-A`
: Instructs the server to autoconfigure at standard **TCK** rate.

  This parameter is ignored unless the **Start-up Configuration** is set to **None** in the Start-up Options dialog (see *Start-up Options dialog* on page 3-16). All other values for the **Start-up Configuration** override the `-A` parameter.

`config_file`
: Specifies a configuration file to use. The settings in the file override the settings in the Start-up Options dialog.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix D
Processor-specific Information

This appendix documents information specific to particular processors. It includes the following sections:

- *The ARM1020T (Rev 0) processor* on page D-2
- *Intel XScale microarchitecture processors* on page D-3.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D.1 The ARM1020T (Rev 0) processor

The ARM1020T processor includes an enhanced CPU, a vector floating-point coprocessor, and integrated debug logic. For more information on the processor see *ARM1020T (Rev 0) Technical Reference Manual*.

This section describes how to use Multi-ICE to debug programs running on an ARM1020T.

D.1.1 Limitations of the ARM1020T (Rev 0) processor

Multi-ICE does not support the following facilities when connected to an ARM1020T (Rev 0) processor:

- big-endian systems
- synchronized start/stop of multiple processors
- hardware watchpoints.

The processor does not implement writing to memory in debug state when the cache is switched on correctly. This affects several debugger actions, including downloading code and setting software breakpoints. You must therefore obey the following rules:

- Always download code with the cache off.
- Do not set more than four breakpoints in code with the cache on. This leaves two hardware breakpoint units that the debugger can use to control program execution.

If the DCache is switched on when the processor enters debug state, Multi-ICE must clean the DCache. To do this it downloads some code to memory at the configured cache clean code address and runs it. Because the processor does not correctly implement writing memory in debug state when the cache is switched on, Multi-ICE might not be able to download this code. To ensure that it can, you must:

- ensure that there is memory at the cache clean code address
- the memory is not in either the ICache or the DCache at any time the processor enters debug state.

You can do this by either:

- specifying a cache clean address region that is never referenced by the application and never referenced by the debugger or operating system (for example, to check memory integrity)
- marking the memory as non-cachable in the MMU tables.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D.2 Intel XScale microarchitecture processors

Intel XScale microarchitecture processors are the successors to the Intel StrongARM, and include an enhanced CPU with several coprocessors. For more information on these processors see *The Intel® XScale™ Core Developer's Manual*.

This section includes information on how to use Multi-ICE to debug programs running on Intel XScale microarchitecture processors. It is divided into the following sections:

- *Behavior on system reset*
- *Debug mode* on page D-5
- *Performance counters* on page D-6
- *Coprocessors* on page D-7
- *Debug handler firmware support* on page D-7
- *Summary* on page D-9.

D.2.1 Behavior on system reset

The XScale processor debug architecture differs significantly from that of other processors based on the ARM architecture. It enters debug state, for example when a breakpoint is hit, by branching to a debugger-specific handler in a new processor mode called Debug. This handler is stored in a special part of the processor instruction cache, and is usually put there when the processor is held in reset.

Multi-ICE is limited to the following ways that it can connect to an XScale microarchitecture processor:

- Multi-ICE uses **nSRST** to hold the processor in reset while downloading the debug handler. It then asserts a debug signal before letting the processor come out of reset, essentially causing it to branch to the reset handler.
- The debug handler remains in the cache if the following are all true:
  - Multi-ICE has previously downloaded the debug handler
  - the **Flush debug handler cache if running on exit** box is not checked in the **Processor Settings** tab of the Multi-ICE configuration dialog (see *Processor Settings Tab* on page 4-14)
  - if the **Leave processor in Monitor mode on exit** option is selected in the **Processor Settings** tab, the processor has not been reset
  - the processor has not been power cycled.

  In such cases, Multi-ICE does not have to download the debug handler again. You can therefore reconnect to a debug session. This is supported whether or not the processor was left running when the previous debug session was terminated.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- The reset handler in the system firmware can support Intel hot-debug. In this case, a processor can be booted from power-on (or any other kind of) reset, allowed to run, and a debugger can later attach to the processor without having to reset it again. Because the system is not reset, its state at the point of connection is not lost. Contact Intel for more information on using hot-debug.

Note

Multi-ICE Version 2.0 only supports connection using a reset.

If you want to reset the processor to connect to it:

- Deselect the **Enable hot-debug** checkbox on the **Processor Settings** tab on the Multi-ICE configuration dialog.
- The signals **nTRST** and **nSRST** must be connected on the target board.
- You cannot perform post-mortem debugging, other than to examine the state of static memory. The contents of dynamic memory might not survive system reset.
- You cannot debug multiple XScale processors connected together, nor can you debug an XScale and ARM-designed processor unless you connect a debugger to the XScale processor first.

If you intend to reconnect to a previously set up debug handler installed either through the use of hot-debug enabled system firmware or from a previous Multi-ICE session:

- You must ensure that the application does not make any SWI calls while the debugger is disconnected. If the application does, it hangs, and the debugger cannot then reconnect without resetting the processor.
- The XScale processor is always stopped when the debugger connects. It is not possible to do nonstop debugging with an XScale processor.
- When you reconnect, Multi-ICE discovers the address the debug handler runs at. If this differs from the currently configured address, a message box asks you to accept the existing address (click **Yes**), reload at the new address (click **No**), or to **Cancel** the connection attempt (although the processor is already stopped).
- If Multi-ICE cannot establish a connection with the debug handler, a message box is displayed asking you to confirm connection by resetting the processor. If you click **No**, the connection attempt is aborted.

Use the **Processor Settings** tab on the Multi-ICE configuration dialog to configure the debug handler address in the special cache that is used (see *Processor Settings Tab* on page 4-14).

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D.2.2 Debug mode

The XScale microarchitecture adds a new processor mode, analogous to existing modes such as Undef mode, called Debug mode. The processor enters Debug mode when a debug event occurs, such as:

- the processor encounters a breakpoint
- the debug request signal input is asserted.

The debug handler uses the DCC on the chip to communicate with the host debugger. Because of this, the DCC cannot be used in other ways, for example:

- for DCC semihosting
- to support the ARM RealMonitor debug agent
- with DCC channel viewers.

There is no debug vector. Instead, the reset vector is overloaded so both the normal reset handler and the debug handler use it. There are therefore two cases:

- When a power-on reset occurs, the processor uses the true reset vector, loaded from ROM.
- If a debugger resets the processor, it installs a debug handler while **nSRST** is asserted. When the debugger deasserts **nSRST**, the processor vectors to the debug handler.

The consequence of this overloading is that your application program cannot use a branch to location zero as a way of simulating a hard reset when a debugger is connected. However, because Multi-ICE has control over the processor it can emulate such a simulated reset, so if you force pc to zero within the debugger, Multi-ICE translates the branch to zero into a branch to the reset handler, so running the reset code. However, to do this it must emulate the instruction at the reset vector, and it only does this for the instructions:

- `LDR <rd>, [<rn>, #Immed12]`
- `B` *`loc`*
- `{MOV|MVN} <rd>, <rn>, {<rm> |#Immed}`
- `{ADD|SUB|RSB|AND|EOR|ORR} <rd>, <rn>, {<rm> |#Immed}`

where *`loc`* is an address offset.

The debug version of the reset vector is held in a cache line in a part of the cache called the mini-ICache. The mini-ICache is mapped over the memory starting at `0x0` and at `0xffff0000` and is used when the processor is in debug state.

Cache lines on current XScale processors are big enough to include every exception vector, not just the reset vector. This means that when a debugger is active, other exceptions, for example interrupts, also use the debug vectors, and might therefore fail.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Multi-ICE copies the currently defined vectors from normal memory to the mini-ICache before executing a program, and also before resuming after a single step or breakpoint. This works well provided that the program does not change the vectors itself. However, if a program changes an exception vector and then that exception occurs (for example, with a SWI opcode), the changed vector is ignored and the old value is used instead.

To avoid this:

1. Place a breakpoint between writing to the exception vector address and the first time the exception can happen.
2. Run the program.
3. When the breakpoint is hit, use the debugger **Continue** or **Go** commands to continue program execution. As a result of continuing, Multi-ICE rewrites the mini-ICache vectors from main memory and so exceptions vector to the new handler.

D.2.3 Performance counters

The ADS debuggers enable you to see the performance counters in the Intel XScale processor. However, the XScale processor does not automatically disable its performance counters when it enters debug state. Because using the debug monitor involves executing code, the performance counters increment when in debug state.

The Multi-ICE debug monitor disables the performance counters in debug state, and re-enables them on exit from debug state.

Because the debug handler cannot disable the counters immediately, some effect of debug state is seen on the counters whenever the processor enters and leaves debug state. For example, if a performance counter is configured to count the number of instructions executed, single stepping a single instruction counts 13 instructions executed, rather than the one expected. This is because 12 instructions are executed in entering debug state and disabling the counter, and later in re-enabling the counters and exiting debug state, as well as the single instruction stepped.

When starting execution from a breakpointed instruction, the debugger first single-steps past the breakpointed instruction, and then resumes execution of the program. Therefore when running from one breakpointed instruction to another breakpoint, the effect is doubled, and 26 additional instructions are recorded. When using semihosting this effect is seen for every semihosting SWI instruction executed.

Note

The figures given here are implementation-dependent, and might vary with different versions of Multi-ICE or XScale microarchitecture processor.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D.2.4 Coprocessors

The XScale microarchitecture processors include several coprocessors:

- coprocessor 0, used for signal processing
- coprocessor 14, used for debugging
- coprocessor 15, used for system control.

The real coprocessor 0 registers found on XScale microarchitecture processors replace the emulation of the EmbeddedICE registers as coprocessor 0 found in ADW.

You must take care when using coprocessors 14 and 15 because the valid instructions and registers are not the same as those on ARM processors.

D.2.5 Debug handler firmware support

The code in Example D-1 is an example of a reset handler that enables hot-debug when it is included in your target firmware, and you are using Multi-ICE Version 2.1 or later.

Example D-1 Example hot-debug firmware

```
reset_handler_start
        ; The reset handler must first check whether this is a debug exception,
        ; or a real RESET event.
        ; NOTE: r13 is the only safe scratch register to use:
        ;
        ;   - for a RESET, any register can be used
        ;
        ;   - for a debug exception, r13 = r13_dbg, and using r13 prevents
        ;     the application registers from being corrupted before the debug
        ;     handler can save.
        MRS     r13, cpsr
        AND     r13, r13, #0x1f
        CMP     r13, #0x15              ; Are we in DBG mode?
        BEQ     dbg_handler_stub        ; if so, go to the dbg handler stub;
        MOV     r13, #0x8000001c        ; otherwise, enable debug, set MOE bits,
        MCR     p14, 0, r13, c10, c0, 0 ; and continue with the reset handler.
        ; Normal reset handler initialization follows code here,
        ; or branch to the reset handler.
        ALIGN   32      ; Align code to a cache line boundary (32 byte aligned).
dbg_handler_stub
        ; First save the state of the IC enable/disable bit in r14_dbg[0].
        MRC     p15, 0, r13, c1, c0, 0
        AND     r13, r13, #0x1000
        ORR     r14, r14, r13, lsr #12
        ; Next, enable the IC.
        MRC     p15, 0, r13, c1, c0, 0
```


Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        ORR     r13, r13, #0x1000
        MCR     p15, 0, r13, c1, c0, 0
        ; Do a sync operation to ensure all outstanding instruction fetches
        ; have completed before continuing.
        ;
        ; The invalidate cache line function serves as a synchronization
        ; operation, and that is why it is used here. The target line is some
        ; scratch address in memory, reserved at the end of this code.
        ADR     r13, line2
        MCR     p15, 0, r13, c7, c5, 1
        ; Invalidate the BTB.
        ;
        ; Make sure that the downloaded vector table does not hit one of the
        ; application's branches that is cached in the BTB, and therefore
        ; branch to the wrong place.
        MCR     p15, 0, r13, c7, c5, 6
        ; Now, send a 'ready for download' message to the debugger, indicating
        ; that the debugger can begin the download.
        ;
        ; NOTE: 'ready for download' = 0x00B00000.
TXloop
        MRC     p14, 0, r15, c14, c0, 0 ; First ensure TX register is available,
        BVS     TXloop                  ; looping if it is not.
        MOV     r13, #0x00B00000        ; Create 'ready for download' message,
        MCR     p14, 0, r13, c8, c0, 0  ; and write it to the TX register
        ; Wait for the debugger to indicate that the download is complete.
RXloop
        MRC     p14, 0, r15, c14, c0, 0 ; Wait for data from the debugger in
        BPL     RXloop                  ; the RX register, looping if none.
        ; Before reading the RX register to get the address to branch to,
        ; restore the state of the IC (saved in DBG_r14[0]) to the value that
        ; it had at the start of the debug handler stub.
        ;
        ; NOTE: the state of the IC must be restored before reading the RX
        ; register, because r13 is the only usable scratch register.
        MRC     p15, 0, r13, c1, c0, 0
        ; First, check r14_dbg[0] to see if the IC was enabled or disabled.
        TST     r14, #0x1
        ; If the IC was previously disabled, then disable it now.
        ;
        ; (Otherwise, there is no need to change the state, because the IC is
        ; already enabled.)
        BICEQ   r13, r13, #0x1000
        MRC     p15, 0, r13, c1, c0, 0
        ; Now r13 can be used to read the RX register and get the target
        ; address to branch to.
        MRC     p14, 0, r13, c9, c0, 0 ; Read the RX register, and
        MOV     pc, r13                ; branch to the downloaded address.
        ; Scratch memory space used by the invalidate IC line function above.
        ALIGN   32                     ; Make sure it starts at a cache line
```


Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
                                        ; boundary, so nothing else is affected.
line2
        SPACE   32                      ; Allocate 32 bytes of zeroed memory.
```

There are several restrictions to debug handlers:

- The code must be in a cachable region of memory. If the code is noncachable, then it might conflict for some of the hardware used by Multi-ICE when it downloads the new monitor.
- The entry point `reset_handler_start` is entered, in the case of an actual debug exception, in Debug Mode. The only register that might be corrupted is `R13` because this does not form any part of the state of the application being debugged. However, the use of `R13` is itself restricted by the XScale microarchitecture. No registers other than `R15` can be changed in getting to `reset_handler_start`.
- The scratch space `line2` must be a region of 32 bytes (1 cache line) aligned to a 32-byte boundary, in cachable memory.
- The normal reset handler must also enable debug and signal hot-debug support in the way shown. That is bit 31 and bits [5:3] of the `DCSR` must all be set.
- The word the target sends up to the debugger must be the value `0x00B00000`. Multi-ICE tests for this value and only allows hot-debug if this value is received.
- To ensure correct operation of hot-debug, you must set appropriate options in the **Processor Settings** tab of the Multi-ICE configuration dialog (see *Processor Settings Tab* on page 4-14).

D.2.6 Summary

This is a summary of the things you must, and must not do when using the XScale microarchitecture processors with Multi-ICE.

Things you must do

You must:

- wire up the reset signals to the Multi-ICE interface unit
- put a breakpoint between writing to an exception vector or changing the memory map and the first time the exception can happen
- use `LDR pc, [`*`loc`*`]` or `B` *`loc`* instructions in the exception vectors.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Things you must not do

You must not:

- write programs that branch to address `0x0`, or a branch to addresses `0xFFFF0000`, for example to simulate a reset
- use the 2KB of address space from addresses `0x0` or `0XFFFF0000` to store code
- expect coprocessor zero to show you EmbeddedICE logic registers.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix E
CP15 Register Mapping

This appendix contains information about the system coprocessor register mapping. It contains the following sections:

- *About register mapping* on page E-2
- *ARM710T processor registers* on page E-3
- *ARM720T processor registers* on page E-4
- *ARM740T processor registers* on page E-5
- *ARM920T and ARM922T processor registers* on page E-6
- *ARM925T processor registers* on page E-10
- *ARM926EJ-S processor registers* on page E-14
- *ARM940T processor registers* on page E-18
- *ARM946E-S processor registers* on page E-21
- *ARM1020T and ARM10200T processor registers* on page E-24
- *XScale microarchitecture processor registers* on page E-28.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.1 About register mapping

From ADS v1.1 onwards, AXD can accept descriptions of the target, and display the coprocessor registers named and decoded into bitfields.

However, ADW and ADU only support the display of coprocessor registers in a list format where each entry corresponds to one of the 16 standard registers. This leaves a problem because the standard register numbers in CP15 are used for more than one function. For example, on the ARM710T device, CP15 r8 is used to either flush the entire TLB or flush a single TLB entry. To work around this, Multi-ICE uses two debugger internal variables to specify the actual coprocessor register that is accessed when a coprocessor register is read.

Only the standard registers appear in the coprocessor window. The values of debugger internal variables defined by Multi-ICE and the data value written to a register are used to determine the exact meaning of each coprocessor register access.

The extra debugger internal variables that have been defined are:

- `cp15_cache_selected`
- `cp15_current_memory_area`.

See *Debugger internal variables* on page 4-36.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.2 ARM710T processor registers

Table E-1 describes the ARM710T processor registers.

Table E-1 ARM710T processor registers

Register	Description	Access	Data
c0	ID register	Read-only	-
c1	Control register	Read/write	Config value
c2	Translation Table Base register	Read/write	Base address
c3	Domain Access Control register	Read/write	Domain value
c5	Fault Status register	Read/write	Fault value
c6	Fault Address register	Read/write	Fault address
c7	Cache Operations: • Invalidate ID Cache	Write-only	 SBZ
c8	TLB Operations: • Invalidate whole TLB • Invalidate Single Entry	Write-only	 SBZ Virtual address

The encodings to read or write the registers are as follows:

c0 to c3, c5, c6

All data reads and writes occur as expected.

c7 Writing any value invalidates the ID Cache.

c8 Writing 0 invalidates the whole TLB.

Writing an *address* with bit 0 set to 1 invalidates the TLB entry for that *address*.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.3 ARM720T processor registers

Table E-2 describes the ARM720T processor registers.

Table E-2 ARM720T processor registers

Register	Description	Access	Data
c0	ID register	Read-only	-
c1	Control register	Read/write	Configuration data
c2	Translation Table Base register	Read/write	Base address
c3	Domain Access Control register	Read/write	Domain value
c5	Fault Status register	Read/write	Fault value
c6	Fault Address register	Read/write	Fault address
c7	Cache operations: • Invalidate ID Cache	Write-only	 SBZ
c8	TLB operations: • Invalidate whole TLB • Invalidate Single Entry	Write-only	 SBZ Virtual address
c13	Process ID register (WinCE)	Read/write	Process ID

The encodings to read or write the registers are as follows:

c0 to c3, c5, c6

All data reads and writes occur as expected.

c7 Writing any value invalidates the ID Cache.

c8 Writing 0 invalidates the whole TLB.

Writing an *address* with bit 0 set to 1 invalidates the TLB entry for that *address*.

c13 All data reads and writes occur as expected.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.4 ARM740T processor registers

Table E-3 describes the ARM740T processor registers.

Table E-3 ARM740T processor registers

Register	Description	Access	Data
c0	ID register	Read-only	-
c1	Control register	Read/write	Configuration data
c2	Cache control	Read/write	Cache control flags
c3	Bufferable control	Read/write	Buffer control flags
c5	Memory protection	Read/write	Memory protection data
c6	Memory area definition: • Memory Region 0 to 7	Read/write	 Base, size, and enable
c7	Cache operations: • Invalidate ID Cache	Write-only	 SBZ

The encodings to read or write the registers are as follows:

c0 to c3, c5 All data reads and writes occur as expected.

c6 The data that is read or written is a memory area definition, and consists of a base address, a size value, and an enable flag. The memory area is specified by the `cp15_current_memory_area` variable.

c7 Writing any value invalidates the ID Cache.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.5 ARM920T and ARM922T processor registers

Table E-4 describes the ARM920T and ARM922T processor registers.

Table E-4 ARM920T and ARM922T processor registers

Register	Description	Access	Data
c0	ID register	Read-only	ID information
		Read-only	Cache configuration[a]
c1	Control register	Read/write	Configuration flags
c2	Translation Table Base	Read/write	Translation table base
c3	Domain Access Control	Read/write	Access flags
c5	Fault Status register	Read/write	Status info
	Prefetch Fault Status register	Read/write	Status info[b]
c6	Fault Address register	Read/write	Fault address
c7	Cache operations:	Write-only	
	• Invalidate ICache and DCache		SBZ
	• Invalidate ICache		SBZ
	• Invalidate I single entry (VA)		VA
	• Prefetch ICache Line		VA
	• Invalidate DCache		SBZ
	• Invalidate D single entry (VA)		VA
	• Clean D single entry (VA)		VA
	• Clean and Invalidate D single entry		VA
	• Clean D single entry (index)		Index and segment
	• Clean and Invalidate D single entry (index)		Index and segment
	• Drain Write Buffer		SBZ
c8	TLB operations:	Read/write	
	• Invalidate ITLB and DTLB		SBZ
	• Invalidate ITLB		SBZ
	• Invalidate ITLB single entry (VA)		VA
	• Invalidate DTLB		SBZ
	• Invalidate DTLB single entry (VA)		VA
c9	Cache lockdown control:	Read/write	
	• Data Lockdown Control		Base and victim
	• Instruction Lockdown Control		Base and victim

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-4 ARM920T and ARM922T processor registers (continued)

Register	Description	Access	Data
c10	TLB lockdown control: • Data Lockdown Base • Instruction Lockdown Base	Read/write	 Base and victim Base and victim
c13	Process ID	Read/write	Process ID
c15	Test and Debug register	Read/write	DCAM and ICAM flags

a. Revision 1 onwards.
b. Revision 1 onwards.

The encodings to read or write the registers are as follows:

c0 A data read with `cp15_cache_selected` = 0 accesses the ID register.

A data read with `cp15_cache_selected` = 1 accesses the Cache Configuration register.

c1, c2, c3 All data reads and writes occur as expected.

c5 A data read with `cp15_cache_selected` = 0 accesses the FSR (Data aborts).

A data read with `cp15_cache_selected` = 1 accesses the PFSR (Prefetch aborts).

Note

The PFSR only exists on revision 1 of the processor onwards.

c6 All data reads and writes occur as expected.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

c7 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [2:0] of the data that is written, as shown in Table E-5.

Table E-5 ARM920T and ARM922T cp15 register 7 accesses

cp15_cache_selected	Bit 2	Bit 1	Bit 0	Purpose
1	0	0	0	Invalidate ICache and DCache
1	0	0	1	Invalidate ICache
1	0	1	0	Invalidate I single entry (VA)
1	0	1	1	Prefetch ICache Line
0	0	0	0	Invalidate DCache
0	0	0	1	Invalidate D single entry
0	0	1	0	Clean D single entry (VA)
0	0	1	1	Clean and Invalidate D single entry (VA)
0	1	0	0	Clean D single entry (index)
0	1	0	1	Clean and Invalidate D single entry
0	1	1	0	Drain Write Buffer

The encoded function uses bits [31:3] of the data that is written, with bits [2:0] cleared.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

c8 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [1:0] of the data that is written, as shown in Table E-6.

Table E-6 ARM920T and ARM922T cp15 register 8 accesses

cp15_cache_selected	Bit 1	Bit 0	Purpose
1	0	0	Invalidate ITLB and DTLB
1	0	1	Invalidate ITLB
1	1	0	Invalidate ITLB single entry (VA)
0	0	0	Invalidate DTLB
0	0	1	Invalidate DTLB single entry (VA)

The encoded function uses bits [31:2] of the data that is written, with bits [1:0] cleared.

c9 A data read or write with `cp15_cache_selected` = 0 accesses the Data Cache Lockdown Base.

A data read with `cp15_cache_selected` = 1 accesses the Instruction Cache Lockdown Base.

c10 A data read or write with `cp15_cache_selected` = 0 accesses the Data TLB Lockdown register.

A data read with `cp15_cache_selected` = 1 accesses the Instruction TLB Lockdown register.

c13 All data reads and writes occur as expected.

Note

The `cp15_current_memory_area` variable is not used with the ARM920T processor.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.6 ARM925T processor registers

Table E-7 describes the ARM925T processor registers.

Table E-7 ARM925T processor registers

Register	Description	Access	Data
c0	ID register	Read-only	ID information
		Read-only	Cache configuration[a]
c1	Control register	Read/write	Configuration flags
c2	Translation Table Base	Read/write	Translation table base
c3	Domain Access Control	Read/write	Access flags
c5	Fault Status register	Read/write	Status info
c6	Fault Address register	Read/write	Fault address
c7	Cache operations:	Write-only	
	• Invalidate ICache and DCache		SBZ
	• Invalidate ICache		SBZ
	• Invalidate I single entry (VA)		VA
	• Prefetch ICache Line		VA
	• Invalidate DCache		SBZ
	• Invalidate D single entry (VA)		VA
	• Invalidate D single entry (index)		Index and segment
	• Clean entire DCache		SBZ
	• Clean D single entry (VA)		VA
	• Clean and Invalidate D single entry		VA
	• Clean D single entry (index)		Index and segment
	• Clean and Invalidate D single entry (index)		Index and segment
	• Drain Write Buffer		SBZ
c8	TLB operations:	Read/write	
	• Invalidate ITLB and DTLB		SBZ
	• Invalidate ITLB		SBZ
	• Invalidate ITLB single entry (VA)		VA
	• Invalidate DTLB		SBZ
	• Invalidate DTLB single entry (VA)		VA

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-7 ARM925T processor registers (continued)

Register	Description	Access	Data
c10	TLB lockdown control: • Data Lockdown Base • Instruction Lockdown Base	Read/write	 Base and victim Base and victim
c13	Process ID	Read/write	Process ID
c15	TI specific registers: • I-max • I-min • Thread ID • ARM925T status register	Read/write	Configuration bits I-max I-min Thread ID Status bits

a. Revision 1 onwards.

The encodings to read or write the registers are as follows:

c0 A data read with `cp15_cache_selected` = 0 accesses the ID register.

A data read with `cp15_cache_selected` = 1 accesses the Cache Configuration register.

c1, c2, c3 All data reads and writes occur as expected.

c5 All data reads and writes access the FSR, which only records Data Aborts. Prefetch Aborts (caused by faulting an instruction access) are not recorded.

c6 All data reads and writes access the FAR, which only records Data Aborts. Prefetch Aborts (caused by faulting an instruction access) are not recorded.

c7 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [2:0] of the data that is written, as shown in Table E-8.

Table E-8 ARM925T cp15 register 7 accesses

cp15_cache_selected	Bit 2	Bit 1	Bit 0	Purpose
1	0	0	0	Invalidate ICache and DCache
1	0	0	1	Invalidate ICache
1	0	1	0	Invalidate I single entry (VA)

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-8 ARM925T cp15 register 7 accesses (continued)

cp15_cache_selected	Bit 2	Bit 1	Bit 0	Purpose
1	0	1	1	Prefetch ICache Line (VA)
1	1	0	0	Drain Write Buffer
0	0	0	0	Invalidate DCache
0	0	0	1	Invalidate D single entry
0	0	1	0	Invalidate D single entry (index)
0	0	1	1	Clean entire DCache
0	1	0	0	Clean D single entry (VA)
0	1	0	1	Clean and Invalidate D single entry (VA)
0	1	1	0	Clean D single entry (index)
0	1	1	1	Clean and Invalidate D single entry (index)

The encoded function uses bits [31:3] of the data that is written, with bits [2:0] cleared.

c8 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [1:0] of the data that is written, as shown in Table E-9.

Table E-9 ARM925T cp15 register 8 accesses

cp15_cache_selected	Bit 1	Bit 0	Purpose
1	0	0	Invalidate ITLB and DTLB
1	0	1	Invalidate ITLB
1	1	0	Invalidate ITLB single entry (VA)
0	0	0	Invalidate DTLB
0	0	1	Invalidate DTLB single entry (VA)

The encoded function uses bits [31:2] of the data that is written, with bits [1:0] cleared.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

c10 A data read or write with `cp15_cache_selected` = 0 accesses the Data TLB Lockdown register.

A data read with `cp15_cache_selected` = 1 accesses the Instruction TLB Lockdown register.

c13 All data reads and writes occur as expected.

c15 The function performed is determined by the value of the `cp15_current_memory_area` variable (although the functions are not related to memory areas), as shown in Table E-10.

Table E-10 ARM925T cp15 register 7 accesses

cp15_current_memory_area	Access	Purpose
0	Read/write	ARM925T configuration register
1	Read/write	I-max
2	Read/write	I-min
3	Read/write	Thread ID
4	Read only	ARM925T status register

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.7 ARM926EJ-S processor registers

Table E-11 describes the ARM926EJ-S processor registers.

Table E-11 ARM926EJ-S processor registers

Register	Description	Access	Data
c0	ID register	Read-only Read-only Read-only	ID information Cache configuration Tightly coupled memory information
c1	Control register	Read/write	Configuration flags
c2	Translation Table Base	Read/write	Translation table base
c3	Domain Access Control	Read/write	Access flags
c5	Fault Status register	Read/write	Status info
c6	Fault Address register	Read/write	Fault address
c7	Cache operations: • Invalidate ICache and DCache • Invalidate ICache • Invalidate I single entry (VA) • Invalidate I single entry (set/way) • Prefetch ICache Line • Invalidate DCache • Invalidate D single entry (VA) • Invalidate D single entry (set/way) • Test and clean DCache • Clean D single entry (VA) • Clean D single entry (set/way) • Test, clean, and invalidate DCache • Clean and invalidate D single entry (VA) • Clean and invalidate D single entry (set/way) • Drain Write Buffer	Write-only	 SBZ SBZ VA Set and way SBZ SBZ VA Set and way SBZ VA Set and way SBZ VA Set and way SBZ

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-11 ARM926EJ-S processor registers (continued)

Register	Description	Access	Data
c8	TLB operations:	Read/write	
	• Invalidate ITLB and DTLB		SBZ
	• Invalidate ITLB and DTLB single entry (VA)		VA
	• Invalidate ITLB		SBZ
	• Invalidate ITLB single entry (VA)		VA
	• Invalidate DTLB		SBZ
	• Invalidate DTLB single entry (VA)		VA
c9	Lockdown control:	Read/write	
	• Data Lockdown Control		DCtrl value
	• Instruction Lockdown Control		ICtrl value
	• Tightly Coupled DMemory Control		DMemory value
	• Tightly Coupled IMemory Control		IMemory value
c10	TLB lockdown control	Read/write	Base and victim
c13	Process identifiers:	Read/write	
	• FCSE PID		FCSE process ID
	• Context ID		ETM context ID
c15	Test and Debug register:	Read/write	
	• Trace Control register		Control flags
	• Memory Region Remap register		Remap information

The encodings to read or write the registers are as follows:

c0 A data read with `cp15_cache_selected` = 0 accesses the ID register.

A data read with `cp15_cache_selected` = 1 accesses the Cache Configuration register.

A data read with `cp15_cache_selected` = 2 accesses the Tightly coupled memory information register.

c1, c2, c3

All data reads and writes occur as expected.

c5 A data read with `cp15_cache_selected` = 0 accesses the Data side FSR.

A data read with `cp15_cache_selected` = 1 accesses the Instruction side FSR.

c6 All data reads and writes occur as expected.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

c7 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [2:0] of the data that is written, as shown in Table E-12.

Table E-12 ARM926EJ-S cp15 register 7 accesses

cp15_cache_selected	Bit 2	Bit 1	Bit 0	Purpose
2	0	0	0	Test, clean, and invalidate DCache
2	0	0	1	Clean and invalidate D single entry (VA)
2	0	1	1	Clean and invalidate D single entry (set/way)
2	1	1	0	Drain Write Buffer
1	0	0	0	Invalidate ICache and DCache
1	0	0	1	Invalidate ICache
1	0	1	0	Invalidate I single entry (VA)
1	0	1	1	Invalidate I single entry (set/way)
1	1	0	0	Prefetch ICache Line
1	1	0	1	Drain Write Buffer
0	0	0	0	Invalidate DCache
0	0	0	1	Invalidate D single entry (VA)
0	0	1	0	Invalidate D single entry (set/way)
0	0	1	1	Test and clean DCache
0	0	1	0	Clean D single entry (VA)
0	0	1	0	Clean D single entry (set/way)

The encoded function uses bits [31:3] of the data that is written, with bits [2:0] cleared.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

c8 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bit 0 of the data that is written, as shown in Table E-13.

Table E-13 ARM926EJ-S cp15 register 8 accesses

cp15_cache_selected	Bit 0	Purpose
2	0	Invalidate ITLB and DTLB
2	1	Invalidate ITLB and DTLB single entry (VA)
1	0	Invalidate ITLB
1	1	Invalidate ITLB single entry (VA)
0	0	Invalidate DTLB
0	1	Invalidate DTLB single entry (VA)

The encoded function uses bits [31:1] of the data that is written, with bit 0 cleared.

c9 A data read or write with `cp15_cache_selected` = 0 accesses the Data Lockdown control.

A data read with `cp15_cache_selected` = 1 accesses the Instruction Lockdown control.

A data read or write with `cp15_cache_selected` = 2 accesses the Tightly Coupled DMemory control.

A data read with `cp15_cache_selected` = 3 accesses the Tightly Coupled IMemory control.

c10 All data reads and writes occur as expected.

c13 A data read or write with `cp15_cache_selected` = 0 accesses the FCSE process ID.

A data read with `cp15_cache_selected` = 1 accesses the ETM context ID.

c15 A data read or write with `cp15_cache_selected` = 0 accesses the Trace Control register.

A data read with `cp15_cache_selected` = 1 accesses the Memory Region Remap register.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.8 ARM940T processor registers

Table E-14 describes the ARM940T processor registers.

Table E-14 ARM940T processor registers

Register	Description	Access	Data
c0	ID register	Read-only	ID information
		Read-only	Cache configuration[a]
c1	Control register	Read/write	Configuration flags
c2	Cache control:	Read/write	
	• Data Cache Control		DCache control flags
	• Instruction Cache Control		ICache control flags
c3	Bufferable Control	Read/write	DBuffer control flags
c5	Memory protection:	Read/write	
	• Data Cache Control		DCache protection flags
	• Instruction Cache Control		ICache protection flags
c6	Memory Area Definition:	Read/write	
	• D memory region 0 to 7		Base, size, and enable
	• I memory region 0 to 7		Base, size, and enable
c7	Cache operations:	Write-only	
	• Flush ICache		SBZ
	• Flush ICache Single Entry		Index and segment
	• Flush DCache		SBZ
	• Flush DCache Single Entry		Index and segment
	• Clean DCache Entry		Index and segment
	• Prefetch ICache Line		Address
	• Clean and Flush DCache entry		Index and segment
	• Drain Write Buffer		SBZ
c9	Cache lockdown control:	Read/write	
	• Data Lockdown Control		D-control value
	• Instruction Lockdown Control		I-control value
c15	Test and Debug register	Read/write	Map I or D CAM flags

a. Revision 1 onwards.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The encodings to read or write the registers are as follows:

c0 A data read with `cp15_cache_selected` = 0 accesses the ID register.

From revision 1 onwards, a data read with `cp15_cache_selected` = 1 accesses the Cache Configuration register.

c1 All data reads and writes occur as expected.

c2 A data read or write with `cp15_cache_selected = 0` accesses the DCache bits.

A data read or write with `cp15_cache_selected` = 1 accesses the ICache bits.

c3 All data reads and writes occur as expected.

c5 A data read or write with `cp15_cache_selected` = 0 accesses the data protection access permissions.

A data read or write with `cp15_cache_selected` = 1 accesses the instruction protection access permissions.

c6 The data that is read or written is a memory area definition, and consists of a base address, a size value, and an enable flag. The memory area is specified by the `cp15_current_memory_area` variable, and the `cp15_cache_selected` variable selects between the D area (when 0) and the I area (when 1).

c7 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [1:0] of the data that is written, as shown in Table E-15.

Table E-15 ARM940T cp15 register 7 accesses

cp15_cache_selected	Bit 1	Bit 0	Purpose
0	0	0	Flush DCache
0	0	1	Flush 1 entry DCache
0	1	0	Clean DCache entry
0	1	1	Clean and flush DCache entry
1	0	0	Flush ICache

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-15 ARM940T cp15 register 7 accesses (continued)

cp15_cache_selected	Bit 1	Bit 0	Purpose
1	0	1	Flush 1 entry ICache
1	1	0	Prefetch ICache cache line
1	1	1	Drain write buffer[a]

a. Revision 1 onwards.

The encoded function uses bits [31:2] of the data that is written, with bits [1:0] cleared. So if `0x80000002` is written to r7, and `cp15_cache_selected` = 1, then the instruction data at `0x80000000` is prefetched into the ICache.

c8 A data read or write with `cp15_cache_selected` = 0 accesses the data lockdown control.

A data read or write with `cp15_cache_selected` = 1 accesses the instruction lockdown control.

c15 All data reads and writes occur as expected.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.9 ARM946E-S processor registers

Table E-16 describes the ARM946E-S processor registers.

Table E-16 ARM946E-S processor registers

Register	Description	Access	Data
c0	ID register	Read-only	ID information
		Read-only	Cache configuration
		Read-only	Tightly coupled memory information
c1	Control register	Read/write	Configuration flags
c2	Cache control:	Read/write	
	• Data Cache Control		DCache control flags
	• Instruction Cache Control		ICache control flags
c3	Bufferable Control	Read/write	DBuffer control flags
c5	Memory protection:	Read/write	
	• Data Cache Control		DCache protection flags
	• Instruction Cache Control		ICache protection flags
c6	Memory Area Definition:	Read/write	
	• D memory region 0 to 7		Base, size, and enable
	• I memory region 0 to 7		Base, size, and enable
c7	Cache operations:	Write-only	
	• Flush ICache		SBZ
	• Flush ICache Entry by VA		VA
	• Prefetch ICache Line		Address
	• Flush DCache		SBZ
	• Flush DCache Entry by VA		VA
	• Clean DCache Entry by VA		VA
	• Clean and Flush DCache entry by VA		VA
	• Clean DCache Entry by index		Index and segment
	• Clean and Flush DCache entry by index		Index and segment

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-16 ARM946E-S processor registers (continued)

Register	Description	Access	Data
c9	Cache lockdown control: • Data Lockdown Control • Instruction Lockdown Control • Tightly coupled D memory control • Tightly coupled I memory control	Read/write	 D-control value I-control value D-mem value I-mem value
c13	Trace process ID register	Read/write	Trace process ID
c15	Test and Debug register	Read/write	Not supported by Multi-ICE

The encodings to read or write the registers are as follows:

c0 A data read with `cp15_cache_selected` = 0 accesses the ID register.

A data read with `cp15_cache_selected` = 1 accesses the Cache Configuration register.

A data read with `cp15_cache_selected` = 2 accesses the Tightly coupled memory information register.

c1 All data reads and writes occur as expected.

c2 A data read or write with `cp15_cache_selected` = 0 or `cp15_cache_selected` = 2 accesses the DCache bits.

A data read or write with `cp15_cache_selected` = 1 or `cp15_cache_selected` = 3 accesses the ICache bits.

c3 All data reads and writes occur as expected.

c5 A data read and write with `cp15_cache_selected` = 0 or `cp15_cache_selected` = 2 accesses the data protection access permissions.

A data read or write with `cp15_cache_selected` = 1 accesses the instruction protection access permissions.

c6 The data that is read or written is a memory area definition, and consists of a base address, a size value, and an enable flag. The memory area is specified by the `cp15_current_memory_area` variable.

Note

Unlike the ARM940T processor, there are not separate I and D versions of these registers.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

c7 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [2:0] of the data that is written, as shown in Table E-17.

Table E-17 ARM946E-S cp15 register 7 accesses

cp15_cache_selected	Bit 2	Bit 1	Bit 0	Purpose
0	0	0	0	Flush DCache
0	0	0	1	Flush DCache entry by VA
0	0	1	0	Clean DCache entry by VA
0	0	1	1	Clean and flush DCache entry by VA
0	1	0	0	Clean DCache Entry by index
0	1	0	1	Clean and flush DCache Entry by index
1	0	0	0	Flush ICache
1	0	0	1	Flush ICache entry by VA
1	0	1	0	Prefetch ICache line

The encoded function uses bits [31:3] of the data that is written, with bits [2:0] cleared. So if `0x80000002` is written to r7, and `cp15_cache_selected` = 1, then the instruction data at `0x80000000` is prefetched into the ICache.

c9 A data read or write with `cp15_cache_selected` = 0 accesses the data lockdown control.

A data read or write with `cp15_cache_selected` = 1 accesses the instruction lockdown control.

A data read or write with `cp15_cache_selected` = 2 accesses the tightly coupled data memory control.

A data read or write with `cp15_cache_selected` = 3 accesses the tightly coupled instruction memory control.

c13 All data reads and writes occur as expected.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.10 ARM1020T and ARM10200T processor registers

Table E-18 describes the ARM1020T and ARM10200T processor registers.

Table E-18 ARM1020T and ARM10200T processor registers

Register	Description	Access	Data
c0	ID register	Read-only	ID information
		Read-only	Cache configuration
c1	Control register	Read/write	Configuration flags
c2	Translation Table Base	Read/write	Translation table base
c3	Domain Access Control	Read/write	Access flags
c5	Fault Status register	Read/write	Status info
c6	Fault Address register	Read/write	Fault address
c7	Cache operations:	Write-only	
	• Invalidate ICache and DCache		SBZ
	• Invalidate ICache		SBZ
	• Invalidate I single entry (VA)		VA
	• Prefetch ICache Line		VA
	• Invalidate DCache		SBZ
	• Invalidate D single entry (VA)		VA
	• Clean D single entry (VA)		VA
	• Clean and Invalidate D single entry		VA
	• Clean D single entry (index)		Index and segment
	• Clean and Invalidate D single entry (index)		Index and segment
	• Drain Write Buffer		SBZ
c8	TLB operations:	Read/write	
	• Invalidate ITLB and DTLB		SBZ
	• Invalidate ITLB		SBZ
	• Invalidate ITLB single entry (VA)		VA
	• Invalidate DTLB		SBZ
	• Invalidate DTLB single entry (VA)		VA
c9	Cache lockdown control:	Read/write	
	• Data Lockdown base		Base and victim
	• Instruction Lockdown base		Base and victim

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-18 ARM1020T and ARM10200T processor registers (continued)

Register	Description	Access	Data
c10	TLB lockdown control: • Data Lockdown base • Instruction Lockdown base	Read/write	 Base and victim Base and victim
c13	Process ID	Read/write	Process ID
c15	Test and Debug register	Read/write	Not supported by Multi-ICE

The encodings to read or write the registers are as follows:

c0 A data read with `cp15_cache_selected` = 0 accesses the ID register.

A data read with `cp15_cache_selected` = 1 accesses the Cache Configuration register.

c1, c2, c3, c5

All data reads and writes occur as expected.

c6 A data read with `cp15_cache_selected` = 0 accesses the Data side FAR.

A data read with `cp15_cache_selected` = 1 accesses the Instruction side FAR.

c7 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [2:0] of the data that is written, as shown in Table E-19.

Table E-19 ARM1020T and ARM10200T cp15 register 7 accesses

cp15_cache_selected	Bit 2	Bit 1	Bit 0	Purpose
1	0	0	0	Invalidate ICache and DCache
1	0	0	1	Invalidate ICache
1	0	1	0	Invalidate ICache single entry (VA)
1	0	1	1	Prefetch ICache Line
0	0	0	0	Invalidate DCache
0	0	0	1	Invalidate DCache single entry

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table E-19 ARM1020T and ARM10200T cp15 register 7 accesses (continued)

cp15_cache_selected	Bit 2	Bit 1	Bit 0	Purpose
0	0	1	0	Clean DCache single entry (VA)
0	0	1	1	Clean and Invalidate DCache single entry (VA)
0	1	0	0	Clean DCache single entry (index)
0	1	0	1	Clean and Invalidate DCache single entry
0	1	1	0	Drain Write Buffer

The encoded function uses bits [31:3] of the data that is written, with bits [2:0] cleared.

c8 The function performed is determined by the value of the `cp15_cache_selected` variable, and by bits [1:0] of the data that is written, as shown in Table E-20.

Table E-20 ARM1020T and ARM10200T cp15 register 8 accesses

cp15_cache_selected	Bit 1	Bit 0	Purpose
1	0	0	Invalidate ITLB and DTLB
1	0	1	Invalidate ITLB
1	1	0	Invalidate ITLB single entry (VA)
0	0	0	Invalidate DTLB
0	0	1	Invalidate DTLB single entry (VA)

The encoded function uses bits [31:2] of the data that is written, with bits [1:0] cleared.

c9 A data read or write with `cp15_cache_selected` = 0 accesses the Data Cache Lockdown base.

A data read with `cp15_cache_selected` = 1 accesses the Instruction Cache Lockdown base.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

c10 A data read or write with `cp15_cache_selected` = 0 accesses the Data TLB Lockdown register.

A data read with `cp15_cache_selected` = 1 accesses the Instruction TLB Lockdown register.

c13 All data reads and writes occur as expected.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.11 XScale microarchitecture processor registers

XScale microarchitecture processors include several coprocessors that are described in the *The Intel® XScale™ Core Developer's Manual*.

The Multi-ICE support for the coprocessor registers on XScale microarchitecture processors depends on the debugger you are using:

- if you use the ADS v1.1 (or later) AXD then Multi-ICE supports access to all coprocessor registers
- if you are using ADW, or ADS v1.0.1, then only the coprocessor registers that can be accessed using coprocessor instructions with both `CRm` and `opcode_2` equal to zero can be used.

The variables `cp15_cache_selected` and `cp15_current_memory_area` are not used when accessing XScale microarchitecture processors.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix F
JTAG Interface Connections

This appendix describes and illustrates the JTAG pin connections. It contains the following sections:

- *Multi-ICE JTAG interface connections* on page F-2
- *Multi-ICE JTAG port timing characteristics* on page F-5
- *TCK frequencies* on page F-7
- *TCK values* on page F-11.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

F.1 Multi-ICE JTAG interface connections

The JTAG connector is situated at one end of the Multi-ICE interface unit. The connector is a 20-Way *Insulation Displacement Connector* (IDC) keyed box header (2.54mm male) that mates with IDC sockets mounted on a ribbon cable (see Figure F-1).

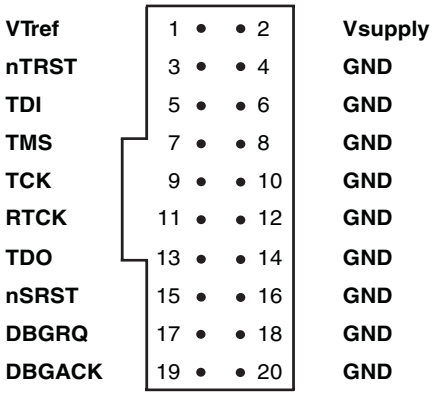


Figure F-1 JTAG pin connections, top view

Note

All **GND** pins must be connected to **0V** on the target board.

F.1.1 JTAG pinouts

Table F-1 on page F-3 shows the JTAG pinouts.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-1 JTAG pinouts

Pin	Signal	I/O	Description
Pin 1	**VTref**	Input	This is the target reference voltage. It indicates that the target has power and it is also used to create the logic-level reference for the input comparators on **TDO** and **RTCK**. It also controls the output logic levels to the target. It is normally fed from $\mathbf{V_{dd}}$ on the target board and might have a series resistor (though this is not recommended).
Pin 2	**Vsupply**	Input	This is the supply voltage to Multi-ICE. It draws its supply current from this pin through a step-up voltage convertor. This is normally fed by the target $\mathbf{V_{dd}}$ which *must not* have a series resistor in the feed to this pin. If the target supply voltage or its current capability is too LOW, this path is broken by an external power jack on the EmbeddedICE adaptor.
Pin 3	**nTRST**	Output	Open collector output from Multi-ICE to the Reset signal on the target JTAG port. This pin must be pulled HIGH on the target to avoid unintentional resets when there is no connection.
Pin 4	**GND**	-	Ground.
Pin 5	**TDI**	Output	Test Data In signal from Multi-ICE to the target JTAG port. It is recommended that this pin is pulled to a defined state.
Pin 6	**GND**	-	Ground.
Pin 7	**TMS**	Output	Test Mode signal from Multi-ICE to the target JTAG port. This pin must be pulled up on the target so that the effect of any spurious **TCK**s when there is no connection is benign.
Pin 8	**GND**	-	Ground.
Pin 9	**TCK**	Output	Test Clock signal from Multi-ICE to the target JTAG port. It is recommended that this pin is pulled to a defined state.
Pin 10	**GND**	-	Ground.
Pin 11	**RTCK**	Input	Return Test Clock signal from the target JTAG port to Multi-ICE. Some targets must synchronize the JTAG inputs to internal clocks. To assist in meeting this requirement, you can use a returned, and retimed, **TCK** to dynamically control the **TCK** rate. Multi-ICE provides Adaptive Clock Timing, which waits for **TCK** changes to be echoed correctly before making further changes. Targets that do not have to process **TCK** can simply ground this pin.
Pin 12	**GND**	-	Ground.
Pin 13	**TDO**	Input	Test Data Out from the target JTAG port to Multi-ICE.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-1 JTAG pinouts (continued)

Pin	Signal	I/O	Description
Pin 14	**GND**	-	Ground.
Pin 15	**nSRST**	Input/output	Open collector output from Multi-ICE to the target system reset. This is also an input to Multi-ICE so that a reset initiated on the target can be reported to the debugger. This pin must be pulled up on the target to avoid unintentional resets when there is no connection.
Pin 16	**GND**	-	Ground.
Pin 17	**DBGRQ**	-	This pin is not connected in the Multi-ICE interface unit. It is reserved for compatibility with other equipment to be used as a debug request signal to the target system.
Pin 18	**GND**	-	Ground.
Pin 19	**DBGACK**	-	This pin is not connected in the Multi-ICE interface unit. It is reserved for compatibility with other equipment to be used as a debug acknowledge signal from the target system.
Pin 20	**GND**	-	Ground.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

F.2 Multi-ICE JTAG port timing characteristics

Figure F-2 and Table F-2 show the timing characteristics of the Multi-ICE unit. These must be considered if you design a target device or board and want to be able to connect Multi-ICE at a particular **TCK** frequency. The characteristics relate to the Multi-ICE hardware. You must consider them in parallel with the characteristics of your target.

In a JTAG device that fully complies to IEEE1149.1, **TDI** and **TMS** are sampled on the rising edge of **TCK**, and **TDO** changes on the falling edge of **TCK**. To take advantage of these properties, Multi-ICE samples **TDO** on the rising edge of **TCK** and changes its **TDI** and **TMS** signals on the falling edge of **TCK**. This means that with a fully compliant target, issues with minimum setup and hold times can always be resolved by simply decreasing the **TCK** frequency, because this increases the separation between signals changing and being sampled.

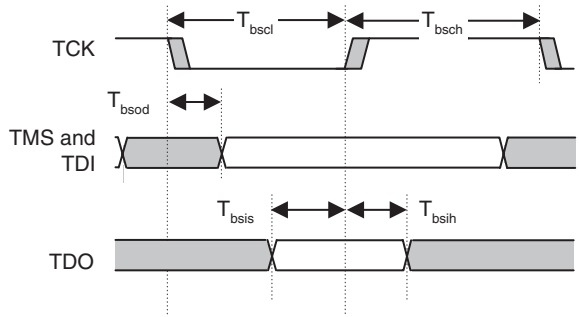


Figure F-2 Multi-ICE JTAG port timing diagram

Table F-2 Multi-ICE IEEE 1149.1 timing requirements

Parameter	Programmed	Min	Max	Description	Note
T_{bscl}	Yes	50ns	204.8µs	**TCK** LOW period	a
T_{bsch}	Yes	50ns	204.8µs	**TCK** HIGH period	b
T_{bsod}	No	-	10ns	**TDI** and **TMS** valid from **TCK** (falling)	c
T_{bsis}	No	27ns	-	**TDO** setup to **TCK** (rising)	d
T_{bsih}	No	10ns	-	**TDO** hold from **TCK** (rising)	-

a. The Multi-ICE server software enables you to change the **TCK** LOW and HIGH periods (see Chapter 3 *Using the Multi-ICE Server*) between the values shown in Table F-2. The other parameters shown in Table F-2 must be considered with the specific values of T_{bscl} and T_{bsch} that you have chosen. The default values for an autoconfigured single-TAP system are, nominally, T_{bscl}=50ns and T_{bsch}=50ns.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

b. The Multi-ICE server software enables you to change the **TCK** LOW and HIGH periods (see Chapter 3 *Using the Multi-ICE Server*) between the values shown in Table F-2 on page F-5. The other parameters shown in Table F-2 on page F-5 must be considered with the specific values of T_{bscl} and T_{bsch} that you have chosen. The default values for an autoconfigured single-TAP system are, nominally, T_{bscl}=50ns and T_{bsch}=50ns.

c. T_{bsod} is the maximum delay between the falling edge of **TCK** and valid levels on the **TDI** and **TMS** Multi-ICE output signals. The target samples these signals on the following rising edge of **TCK** and so the minimum setup time for the target, relative to the rising edge of **TCK**, is $T_{bscl}-T_{bsod}$.

d. T_{bsis} is the minimum setup time for the **TDO** input signal, relative to the rising edge of **TCK** when Multi-ICE samples this signal. The target changes its **TDO** value on the previous falling edge of **TCK** and so the maximum time for the target **TDO** level to become valid, relative to the falling edge of **TCK**, is $T_{bscl}-T_{bsis}$.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

F.3 TCK frequencies

Table F-3 gives the values that must be entered into the **TCK** fields on the JTAG settings dialog for a particular **TCK** frequency. For example, for a 3.33MHz **TCK** rate, use a value of 2 for **TCK** HIGH and **TCK** LOW.

Table F-3 TCK frequencies

Frequency (kHz)	Half-period (ns)	Value	Frequency (kHz)	Half-period (ns)	Value
10000.00	50	0	454.55	1100	21
5000.00	100	1	434.78	1150	22
3333.33	150	2	416.67	1200	23
2500.00	200	3	400.00	1250	24
2000.00	250	4	384.62	1300	25
1666.67	300	5	370.37	1350	26
1428.57	350	6	357.14	1400	27
1250.00	400	7	344.83	1450	28
1111.11	450	8	333.33	1500	29
1000.00	500	9	322.58	1550	30
909.09	550	10	312.50	1600	31
833.33	600	11	294.12	1700	48
769.23	650	12	277.78	1800	49
714.29	700	13	263.16	1900	50
666.67	750	14	250.00	2000	51
625.00	800	15	238.10	2100	52
588.24	850	16	227.27	2200	53
555.56	900	17	217.39	2300	54
526.32	950	18	208.33	2400	55
500.00	1000	19	200.00	2500	56

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-3 TCK frequencies (continued)

Frequency (kHz)	Half-period (ns)	Value	Frequency (kHz)	Half-period (ns)	Value
476.19	1050	20	192.31	2600	57
185.19	2700	58	59.52	8400	116
178.57	2800	59	56.82	8800	117
172.41	2900	60	54.35	9200	118
166.67	3000	61	52.08	9600	119
147.06	3400	80	50.00	10000	120
138.89	3600	81	48.08	10400	121
131.58	3800	82	46.30	10800	122
125.00	4000	83	44.64	11200	123
119.05	4200	84	43.10	11600	124
113.64	4400	85	41.67	12000	125
108.70	4600	86	40.32	12400	126
104.17	4800	87	39.06	12800	127
100.00	5000	88	36.76	13600	144
96.15	5200	89	34.72	14400	145
92.59	5400	90	32.89	15200	146
89.29	5600	91	31.25	16000	147
86.21	5800	92	29.76	16800	148
83.33	6000	93	28.41	17600	149
80.65	6200	94	27.17	18400	150
78.13	6400	95	26.04	19200	151
73.53	6800	112	25.00	20000	152
69.44	7200	113	24.04	20800	153
65.79	7600	114	23.15	21600	154

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-3 TCK frequencies (continued)

Frequency (kHz)	Half-period (ns)	Value	Frequency (kHz)	Half-period (ns)	Value
62.50	8000	115	22.32	22400	155
21.55	23200	156	7.44	67200	212
20.83	24000	157	7.10	70400	213
20.16	24800	158	6.79	73600	214
19.53	25600	159	6.51	76800	215
18.38	27200	176	6.25	80000	216
17.36	28800	177	6.01	83200	217
16.45	30400	178	5.79	86400	218
15.63	32000	179	5.58	89600	219
14.88	33600	180	5.39	92800	220
14.20	35200	181	5.21	96000	221
13.59	36800	182	5.04	99200	222
13.02	38400	183	4.88	102400	223
12.50	40000	184	4.60	108800	240
12.02	41600	185	4.34	115200	241
11.57	43200	186	4.11	121600	242
11.16	44800	187	3.91	128000	243
10.78	46400	188	3.72	134400	244
10.42	48000	189	3.55	140800	245
10.08	49600	190	3.40	147200	246
9.77	51200	191	3.26	153600	247
9.19	54400	208	3.13	160000	248
8.68	57600	209	3.00	166400	249
8.22	60800	210	2.89	172800	250

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-3 TCK frequencies (continued)

Frequency (kHz)	Half-period (ns)	Value	Frequency (kHz)	Half-period (ns)	Value
7.81	64000	211	2.79	179200	251
2.69	185600	252	2.52	198400	254
2.60	192000	253	2.44	204800	255

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

F.4 TCK values

Table F-4 shows the corresponding frequencies for the **TCK** fields on the JTAG settings dialog. As an example, for a value of 4 for **TCK** HIGH and **TCK** LOW, the **TCK** rate is 2MHz.

Table F-4 TCK values

Value	Half-period (ns)	Frequency (kHz)	Value	Half-period (ns)	Frequency (kHz)
0	50	10000.00	21	1100	454.55
1	100	5000.00	22	1150	434.78
2	150	3333.33	23	1200	416.67
3	200	2500.00	24	1250	400.00
4	250	2000.00	25	1300	384.62
5	300	1666.67	26	1350	370.37
6	350	1428.57	27	1400	357.14
7	400	1250.00	28	1450	344.83
8	450	1111.11	29	1500	333.33
9	500	1000.00	30	1550	322.58
10	550	909.09	31	1600	312.50
11	600	833.33	32	100	5000.00
12	650	769.23	33	200	2500.00
13	700	714.29	34	300	1666.67
14	750	666.67	35	400	1250.00
15	800	625.00	36	500	1000.00
16	850	588.24	37	600	833.33
17	900	555.56	38	700	714.29
18	950	526.32	39	800	625.00
19	1000	500.00	40	900	555.56

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-4 TCK values (continued)

Value	Half-period (ns)	Frequency (kHz)	Value	Half-period (ns)	Frequency (kHz)
20	1050	476.19	41	1000	500.00
42	1100	454.55	66	600	833.33
43	1200	416.67	67	800	625.00
44	1300	384.62	68	1000	500.00
45	1400	357.14	69	1200	416.67
46	1500	333.33	70	1400	357.14
47	1600	312.50	71	1600	312.50
48	1700	294.12	72	1800	277.78
49	1800	277.78	73	2000	250.00
50	1900	263.16	74	2200	227.27
51	2000	250.00	75	2400	208.33
52	2100	238.10	76	2600	192.31
53	2200	227.27	77	2800	178.57
54	2300	217.39	78	3000	166.67
55	2400	208.33	79	3200	156.25
56	2500	200.00	80	3400	147.06
57	2600	192.31	81	3600	138.89
58	2700	185.19	82	3800	131.58
59	2800	178.57	83	4000	125.00
60	2900	172.41	84	4200	119.05
61	3000	166.67	85	4400	113.64
62	3100	161.29	86	4600	108.70
63	3200	156.25	87	4800	104.17
64	200	2500.00	88	5000	100.00

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-4 TCK values (continued)

Value	Half-period (ns)	Frequency (kHz)	Value	Half-period (ns)	Frequency (kHz)
65	400	1250.00	89	5200	96.15
90	5400	92.59	114	7600	65.79
91	5600	89.29	115	8000	62.50
92	5800	86.21	116	8400	59.52
93	6000	83.33	117	8800	56.82
94	6200	80.65	118	9200	54.35
95	6400	78.13	119	9600	52.08
96	400	1250.00	120	10000	50.00
97	800	625.00	121	10400	48.08
98	1200	416.67	122	10800	46.30
99	1600	312.50	123	11200	44.64
100	2000	250.00	124	11600	43.10
101	2400	208.33	125	12000	41.67
102	2800	178.57	126	12400	40.32
103	3200	156.25	127	12800	39.06
104	3600	138.89	128	800	625.00
105	4000	125.00	129	1600	312.50
106	4400	113.64	130	2400	208.33
107	4800	104.17	131	3200	156.25
108	5200	96.15	132	4000	125.00
109	5600	89.29	133	4800	104.17
110	6000	83.33	134	5600	89.29
111	6400	78.13	135	6400	78.13
112	6800	73.53	136	7200	69.44

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-4 TCK values (continued)

Value	Half-period (ns)	Frequency (kHz)	Value	Half-period (ns)	Frequency (kHz)
113	7200	69.44	137	8000	62.50
138	8800	56.82	162	4800	104.17
139	9600	52.08	163	6400	78.13
140	10400	48.08	164	8000	62.50
141	11200	44.64	165	9600	52.08
142	12000	41.67	166	11200	44.64
143	12800	39.06	167	12800	39.06
144	13600	36.76	168	14400	34.72
145	14400	34.72	169	16000	31.25
146	15200	32.89	170	17600	28.41
147	16000	31.25	171	19200	26.04
148	16800	29.76	172	20800	24.04
149	17600	28.41	173	22400	22.32
150	18400	27.17	174	24000	20.83
151	19200	26.04	175	25600	19.53
152	20000	25.00	176	27200	18.38
153	20800	24.04	177	28800	17.36
154	21600	23.15	178	30400	16.45
155	22400	22.32	179	32000	15.63
156	23200	21.55	180	33600	14.88
157	24000	20.83	181	35200	14.20
158	24800	20.16	182	36800	13.59
159	25600	19.53	183	38400	13.02
160	1600	312.50	184	40000	12.50

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-4 TCK values (continued)

Value	Half-period (ns)	Frequency (kHz)	Value	Half-period (ns)	Frequency (kHz)
161	3200	156.25	185	41600	12.02
186	43200	11.57	210	60800	8.22
187	44800	11.16	211	64000	7.81
188	46400	10.78	212	67200	7.44
189	48000	10.42	213	70400	7.10
190	49600	10.08	214	73600	6.79
191	51200	9.77	215	76800	6.51
192	3200	156.25	216	80000	6.25
193	6400	78.13	217	83200	6.01
194	9600	52.08	218	86400	5.79
195	12800	39.06	219	89600	5.58
196	16000	31.25	220	92800	5.39
197	19200	26.04	221	96000	5.21
198	22400	22.32	222	99200	5.04
199	25600	19.53	223	102400	4.88
200	28800	17.36	224	6400	78.13
201	32000	15.63	225	12800	39.06
202	35200	14.20	226	19200	26.04
203	38400	13.02	227	25600	19.53
204	41600	12.02	228	32000	15.63
205	44800	11.16	229	38400	13.02
206	48000	10.42	230	44800	11.16
207	51200	9.77	231	51200	9.77
208	54400	9.19	232	57600	8.68

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table F-4 TCK values (continued)

Value	Half-period (ns)	Frequency (kHz)	Value	Half-period (ns)	Frequency (kHz)
209	57600	8.68	233	64000	7.81
234	70400	7.10	245	140800	3.55
235	76800	6.51	246	147200	3.40
236	83200	6.01	247	153600	3.26
237	89600	5.58	248	160000	3.13
238	96000	5.21	249	166400	3.00
239	102400	4.88	250	172800	2.89
240	108800	4.60	251	179200	2.79
241	115200	4.34	252	185600	2.69
242	121600	4.11	253	192000	2.60
243	128000	3.91	254	198400	2.52
244	134400	3.72	255	204800	2.44

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix G
User I/O Connections

This appendix describes and illustrates the additional input and output connections provided in Multi-ICE. It contains the following section:

- *Multi-ICE user I/O pin connections* on page G-2.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

G.1 Multi-ICE user I/O pin connections

This section describes the User I/O connector. It contains the following sections:

- *User input/output pin connections*
- *Input bit logic* on page G-4.

The user (I/O) connector is situated under the removable cover on the Multi-ICE interface unit. The connector is a 20-way header that mates with IDC sockets mounted on a ribbon cable (see Figure G-1).

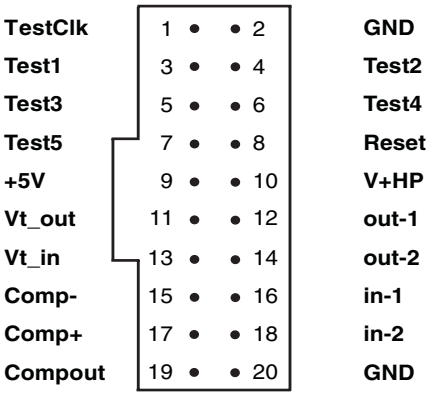


Figure G-1 User I/O pin connections

If you must drive one of the user-defined inputs with a signal operating at the target system logic levels, see the sample circuit in Figure G-2 on page G-4.

G.1.1 User input/output pin connections

Table G-1 shows the user input/output pin connections.

Table G-1 User I/O connections

Pin	Signal	I/O	Description
Pin 1	**TestClk**	-	For production test only. This pin must be left unconnected.
Pin 2	**GND**	-	-
Pin 3	**Test1**	-	For production test only. This pin must be left unconnected.
Pin 4	**Test2**	-	For production test only. This pin must be left unconnected.
Pin 5	**Test3**	-	For production test only. This pin must be left unconnected.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table G-1 User I/O connections (continued)

Pin	Signal	I/O	Description
Pin 6	**Test4**	-	For production test only. This pin must be left unconnected.
Pin 7	**Test5**	-	For production test only. This pin must be left unconnected.
Pin 8	**Reset**	-	For production test only. This pin must be left unconnected.
Pin 9	**+5V**	Output	This is intended for use as a supply for a small amount of external logic circuitry. There is a recommended current limit of 20mA from this pin. You must remember that due to the DC-DC converter, the additional current taken from the target to supply any external logic is approximately `Iout * (5V / target voltage)`.
Pin 10	**V+HP**	-	For production test only. This pin must be left unconnected.
Pin 11	**Vt_out**	-	For production test only. This pin must be left unconnected.
Pin 12	**out-1**	Output	This is a user output bit.
Pin 13	**Vt_in**	-	This is the voltage threshold used for input logic detection, derived from the target logic reference voltage (**VTref** on pin 1 of the target connector). Use this as one of the inputs to the comparator if a target logic level is being monitored.
Pin 14	**out-2**	Output	This is a user output bit.
Pin 15	**Comp-**	Input	This is connected to the inverting input of a spare LM339D comparator for use with the user-defined input/output. There is a 1MΩ pull-down resistor to GND on this pin.
Pin 16	**in-1**	Input	This is a user input bit. This uses 74ACT family input thresholds and has a 10kΩ pull-up to +5V.
Pin 17	**Comp+**	Input	This is connected to the non-inverting input of a spare LM339D comparator for use with the user-defined input/output. There is a 1MΩ pull-up resistor to +5V on this pin.
Pin 18	**in-2**	Input	This is a user input bit. This uses 74ACT family input thresholds and has a 10kΩ pull-up to +5V.
Pin 19	**Compout**	Output	This is connected to the output of a spare LM339D comparator for use with the user-defined input/output. This is an open collector output so it requires a pull-up resistor (user inputs **in-1** and **in-2** already have suitable pull-ups (10kΩ to 5V)).
Pin 20	**GND**	-	-

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

G.1.2 Input bit logic

The user input bits correspond to two TTL logic-level outputs available from the user input/output connector (see *User I/O pin connections* on page G-2). You can use these signals to remotely monitor user logic at the server location.

The Multi-ICE JTAG port automatically adapts its input and output thresholds to the voltage levels in the target system (based on the **VTref** pin). The inputs on the Multi-ICE user I/O connector operate at standard TTL levels. If you must drive one of the user-defined inputs with a signal operating at the target system logic levels, the circuit, as shown in Figure G-2, converts this to TTL levels.

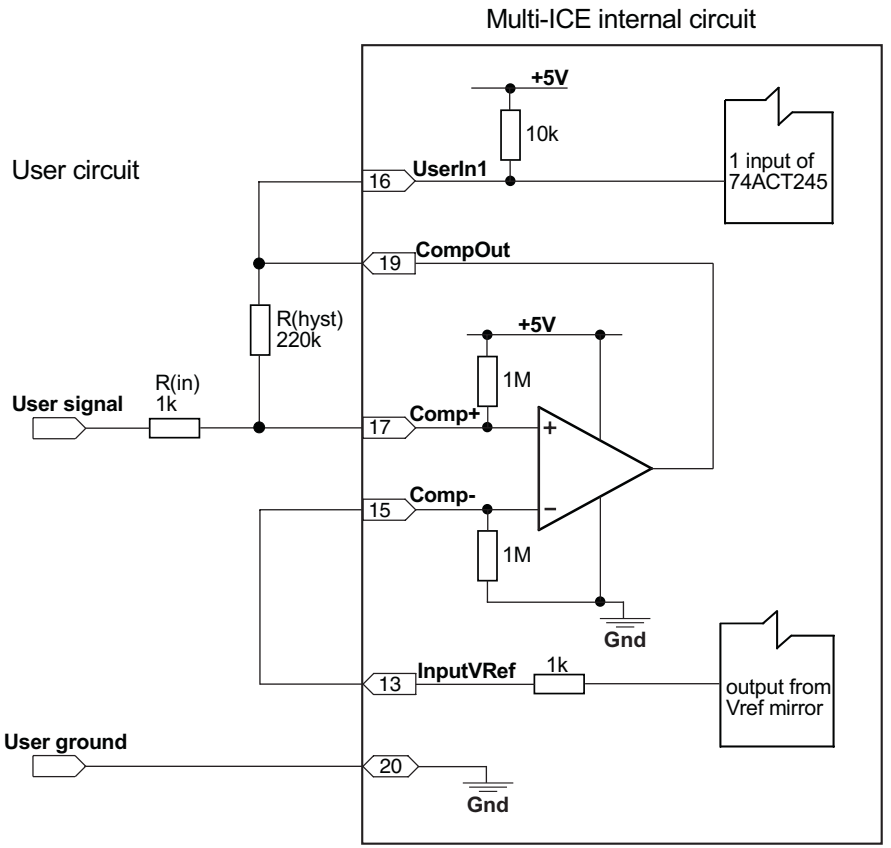


Figure G-2 Converting user-input signals to TTL levels

Pins 15, 17, and 19 are connected to an LM339 type comparator within the Multi-ICE interface unit. The open collector comparator output (pin 19) drives the user input (pin 16), which includes a suitable pull-up resistor. The inverting input to the

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

comparator (pin 15) is driven by an output from the voltage reference mirror circuit (pin 13). The non-inverting input to the comparator (pin 17) is driven by the signal to be monitored using a small series resistor (Rin).

The output of the comparator is also fed back to this input through a large resistor (Rhyst) to provide a small amount of hysteresis (around 20mV with the values shown). A ground reference for the input signal must be connected to pin 20 to provide a more direct return path than through the JTAG connector.

The user input bits are shown at the bottom-right corner of the Multi-ICE server window. Each bit is:

- light green when HIGH
- dark green when LOW.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Glossary

Adaptive clocking A technique in which a clock signal is sent out by Multi-ICE and it waits for the returned clock before generating the next clock pulse. The technique allows the Multi-ICE interface unit to adapt to differing signal drive capabilities and differing cable lengths.

ADS See *ARM Developer Suite*.

ADU See *ARM Debugger for UNIX*.

ADW See *ARM Debugger for Windows*.

Angel Angel is a debug monitor that runs on an ARM-based target and enables you to debug applications.

Application Program Interface A specification for a set of procedures, functions, data structures, and constants that are used to interface two or more software components together. For example, an API between an operating system and the application programs that use it might specify exactly how to read data from a file.

ARM Debugger for UNIX *ARM Debugger for UNIX* (ADU) and *ARM Debugger for Windows* (ADW) are two versions of the same ARM debugger software, running under UNIX or Windows respectively.

ARM Debugger for Windows *ARM Debugger for Windows* (ADW) and *ARM Debugger for UNIX* (ADU) are two versions of the same ARM debugger software, running under Windows or UNIX respectively. This debugger was issued originally as part of the ARM Software Development Toolkit.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ARM Developer Suite A suite of applications, together with supporting documentation and examples, that enable you to write and debug applications for the ARM family of RISC processors.

ARM eXtended Debugger The *ARM eXtended Debugger* (AXD) is the latest debugger software from ARM that enables you to make use of a debug agent in order to examine and control the execution of software running on a debug target. AXD is supplied in both Windows and UNIX versions.

ARMulator ARMulator is an instruction set simulator. It is a collection of modules that simulate the instruction sets and architecture of various ARM processors.

AXD See *ARM eXtended Debugger*.

Big-endian Memory organization where the least significant byte of a word is at a higher address than the most significant byte. See *Little-endian*.

Cache cleaning The process of writing *dirty data* in a cache to main memory.

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

Core Module *See* Integrator

CPU Central Processor Unit.

CPSR Current Program Status Register. See *Program Status Register*.

DCache Data cache.

DLL *See* Dynamic Linked Library

Dirty data When referring to a processor data cache, data that has been written to the cache but has not been written to main memory. Only write-back caches can have dirty data, because a write-through cache writes data to the cache and to main memory simultaneously. The process of writing dirty data to main memory is called *cache cleaning*.

Double word A 64-bit unit of information. Contents are taken as being an unsigned integer unless otherwise stated.

Dynamic Linked Library A collection of programs, any of which can be called when needed by an executing program. A small program that helps a larger program communicate with a device such as a printer or keyboard is often packaged as a DLL.

ECP *See* Enhanced Capability Port.

EmbeddedICE The additional hardware provided by debuggable ARM processors to aid debugging.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Enhanced Capability Port
A standard for parallel ports which enables fast bidirectional data transfers over parallel ports.

See also EPP and IEEE1284.

Enhanced Parallel Port
A standard for parallel ports which enables fast bidirectional data transfers over parallel ports.

See also ECP and IEEE1284.

EPP
See Enhanced Parallel Port.

Environment
The actual hardware and operating system that an application will run on.

ETM
Embedded Trace Macrocell.

External Data Representation
A specification defined by Sun Microsystems describing a way of transferring typed data between computer systems in a system independent manner. Used by Sun RPC.

Flash memory
Nonvolatile memory that is often used to hold application code.

Halfword
A 16-bit unit of information. Contents are taken as being an unsigned integer unless otherwise stated.

Heap
The portion of computer memory that can be used for creating new variables.

Host
A computer which provides data and other services to another computer. *Especially*, a computer providing debugging services to a target being debugged.

ICache
Instruction cache.

ICE
See In-Circuit Emulator.

ICE Extension Unit
A hardware extension to the EmbeddedICE logic that provides more breakpoint units.

ID
Identifier.

IEEE 1149.1
The IEEE Standard which defines TAP. Commonly (but incorrectly) referred to as JTAG.

IEEE 1284
A standard for parallel port interfaces which encompasses ECP but extends it to enable semi-autonomous transfers.

IEU
See ICE Extension Unit.

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

In-Circuit Emulator
A device enabling access to and modification of the signals of a circuit while that circuit is operating.

Instruction Register
When referring to a TAP controller, a register that controls the operation of the TAP.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Integrator An ARM hardware development platform. Core Modules are available that contain the processor and local memory.

IR *See* Instruction Register.

Joint Test Action Group The name of the standards group which created the IEEE 1149.1 specification.

JTAG *See* Joint Test Action Group.

Little-endian Memory organization where the least significant byte of a word is at a lower address than the most significant byte. See also *Big-endian*.

Memory management unit Hardware that controls caches and access permissions to blocks of memory, and translates virtual to physical addresses.

MMU *See* Memory Management Unit.

Multi-ICE Multi-processor EmbeddedICE interface. ARM registered trademark.

nSRST Abbreviation of *System Reset*. The electronic signal which causes the target system other than the TAP controller to be reset. This signal is known as **nSYSRST** in some other manuals.

See also **nTRST**.

nTRST Abbreviation of *TAP Reset*. The electronic signal that causes the target system TAP controller to be reset. This signal is known as **nICERST** in some other manuals.

See also **nSRST**.

Open collector A signal that may be actively driven LOW by one or more drivers, and is otherwise passively pulled HIGH. Also known as a "wired AND" signal.

PID The ARM Platform-Independent Development card, now known as the ARM Development Board.

PIE A platform-independent evaluator card designed and supplied by ARM Limited.

Port mapper A process that enables RPC client processes to contact the RPC server process for a particular RPC service.

Processor Core The part of a microprocessor that reads instructions from memory and executes them, including the instruction fetch unit, arithmetic and logic unit and the register bank. It excludes optional coprocessors, caches, and the memory management unit.

Processor Status Register See *Program Status Register*.

Program image See Image.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Program Status Register
Program Status Register (PSR), containing some information about the current program and some information about the current processor. Often, therefore, also referred to as *Processor Status Register*.

Is also referred to as *Current PSR* (CPSR), to emphasize the distinction between it and the *Saved PSR* (SPSR). The SPSR holds the value the PSR had when the current function was called, and which will be restored when control is returned.

RDI
See Remote Debug Interface.

Remapping
Changing the address of physical memory or devices after the application has started executing. This is typically done to allow RAM to replace ROM once the initialization has been done.

Remote_A
Remote_A is a software protocol converter and configuration interface. It converts between the RDI 1.5 software interface of a debugger and the Angel Debug Protocol used by Angel targets. It can communicate over a serial or Ethernet interface.

Remote Debug Interface
RDI is an open ARM standard procedural interface between a debugger and the debug agent. The widest possible adoption of this standard is encouraged.

Remote Procedure Call
A call to a procedure in a different process. The calling procedure invokes a procedure in a different process that is usually running on a different processor.

RM
RealMonitor.

RPC
See Remote Procedure Call.

RTCK
Returned **TCK**. The signal which enables Adaptive Clocking.

RTOS
Real Time Operating System.

Scan Chain
A group of one or more registers from one or more TAP controllers connected between TDI and TDO, through which test data is shifted.

Semihosting
A mechanism whereby the target communicates I/O requests made in the application code to the host system, rather than attempting to support the I/O itself.

SPSR
Saved Program Status Register. See *Program Status Register*.

SWI
Software Interrupt. An instruction that causes the processor to call a programer-specified subroutine. Used by ARM to handle semihosting.

Synchronous starting
Setting several processors to a particular program location and state, and starting them together.

Synchronous stopping
Stopping several processors in such a way that they stop executing at the same instant.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sun RPC A specific form of RPC defined as a standard by Sun Microsystems that uses the XDR standard and TCP/IP datagrams to communicate between networked computers.

TAP *See* Test Access Port.

TAP Controller Logic on a device which allows access to some or all of that device for test purposes. The circuit functionality is defined in IEEE1149.1.

See also TAP, IEEE1149.1.

TAPOp The name of the interface API between the Multi-ICE Server and its clients.

Target The actual processor (real silicon or simulated) on which the application program is running.

TCK The electronic clock signal which times data on the TAP data lines TMS, TDI, and TDO.

TDI The electronic signal input to a TAP controller from the data source (upstream). Usually this is seen connecting the Multi-ICE Interface Unit to the first TAP controller.

TDO The electronic signal output from a TAP controller to the data sink (downstream). Usually this is seen connecting the last TAP controller to the Multi-ICE Interface Unit.

Test Access Port The port used to access a device's TAP Controller. Comprises TCK, TMS, TDI, TDO and nTRST (optional).

TTL Transistor-transistor logic. A type of logic design in which two bipolar transistors drive the logic output to one or zero. LSI and VLSI logic often used TTL with HIGH logic level approaching +5V and LOW approaching 0V.

Watchpoint A location within the image that will be monitored and that will cause execution to stop when it changes.

Word A 32-bit unit of information. Contents are taken as being an unsigned integer unless otherwise stated.

XDR *See External Data Representation*.

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Index

A

Active server connections 4-12
Adaptive clocking 5-8, 6-3, 6-4
  menu option 3-24
Adaptive JTAG voltage levels 6-13
Adaptor
  TI 14-way to 20-way 2-12
  14-way to 20-way 1-2, 2-9, 4-47, 5-14
ADU 1-3
ADW 1-3, 4-5
  accessing debugger internals 4-50
  Configure Debugger menu item 4-5
  Debugger Configuration diaog 4-6
  error messages 5-16
AFS 1-5
Allow network connections 2-17, 4-12
AMBA 5-9
Angel debug monitor 1-5
ARM
  ADS 1-13, 1-15, 2-2
  ADW 4-2, 4-5
  Angel debug monitor 4-54
  AXD 4-4
  Debuggers 2-2
  Developer Suite 2-2
  Development Board 4-64, 5-8
  Embedded Trace Macrocell 1-13
  Firmware Suite 1-5
  Integrator 5-8
  Integrator board 1-14
  PID board 4-64
  PIE7 board 2-11, 5-8
  PIV7T board 2-11, 5-8
  Processors. *see* Processors
  RealMonitor 1-14, 4-51
  System coprocessor 4-49
  TDT 1-13
ARMulator 4-5
ASIC
  guidelines 6-9
  multiple devices 6-9
  signal drivers 6-15
Automatic
  configuration 3-3
  dialup 2-3
AXD 1-3, 4-4
  Accessing debugger internals 4-50
  Choose Target dialog 4-4
  Configure Target menu item 4-4
  debugging multiple processors 4-27

B

Big endian 4-18, D-2
Boards
  ARM Development 5-8
  ARM Integrator 5-8
  ARM PIE7 2-11, 5-8
  ARM PIV7T 2-11, 5-8
  compatibility 2-11
  Integrator 1-14
  selecting 1-11, 4-21
Box header, JTAG F-2
Breakpoints 4-56, B-2
  exception vectors 5-8
  hardware vs software 4-56

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ROM 4-62
semihosting 4-53
signal 1-4
single-stepping B-3
SWI vector B-2
BREAKPT signal 1-4
Browsing for server 4-11

C

Cable length, JTAG 6-15
Cache
clean code address 4-14
clean data address 4-15
read-ahead memory 4-19
cache
preservation 4-60–4-61
Calculating, JTAG clock 3-22
CE Declaration of Conformity iv
Channel viewers
DLL 4-24
failing 5-9
selecting 4-24
Clocks
adaptive 6-3, 6-4
JTAG speed 3-21
menu option 3-24
setting speed 3-7
synchronizing 3-24, 6-3
Compatibility
boards 2-11
Multi-ICE with debuggers 1-3
Configuration
automatic 3-3, 3-9
manual 3-12
server 2-16
Configuration files
autoconfig.cfg 3-9
creating 3-9
examples A-5
IRlength.arm 5-5, A-3
loading 3-3
userdrvn.txt 3-13, 5-3
Connecting
debugger to Multi-ICE 4-3
Multi-ICE hardware 2-6
Connection
active server 4-12
PCB 6-12
Coprocessor
debug 4-65, D-7
signal processing D-7
system 4-49, D-7, E-1
vector floating-point D-2
0, accessing 4-65
Coprocessor 0 D-7
Coprocessor 15 4-49, D-7
Coprocessors
register display E-2
Cores, CPU
starting and stopping 3-5
Creating
Multi-ICE configuration files 3-9

D

Daisy chaining TAP controllers 6-3
Data Cache uncachable bit 5-13
Data transfer, 4-bit 3-19
DBGACK signal 1-4
DBGEN 5-5, 5-6
DBGRQ signal 1-4
Debug
extensions 1-4
Debug Comms Channel 1-5
viewers *see* Channel Viewers
Debug Communications Channel. *see* Debug Comms Channel.
Debug handler address 4-16
Debugger
ADW 4-2
internal variables 4-36
saving settings 4-26
Debugger variables
see also Variables
Debuggers
connecting to Multi-ICE 4-3
performance 3-30
Debugging
ROM applications 4-62
self-modifying code 4-56
Defining JTAG device interaction 3-28
Desktop Update, Windows 4-3
Devices
execution control 3-27
polling 3-30
unknown 2-16, 5-3
Dialup
automatic 2-3
Digital signal processor 6-2
Disk usage 2-3
Displaying
RPC call information 3-5
DLL files
channel viewer 4-24
Downloading code
user output bits 3-20
Driver
.mul files 5-16

E

ECP parallel port 3-19
Electromagnetic conformity iv
Embedded Trace Macrocell 1-2, 1-13
EmbeddedICE 1-2, 1-4, 4-41, 4-69, 6-17
debug architecture 1-5
Interface Unit 2-13
logic, resetting 4-63, 5-6
resetting logic 6-6
EmbeddedICE/RT logic 1-4
Endian
big 4-18, D-2
little 4-18
EPP parallel port 3-19
Error messages
from ADW 5-16
from Multi-ICE server 5-12
ETM *see* Embedded Trace Macrocell
Examining a running system 4-45
Exception vectors
breakpoints 5-8
setting breakpoints 5-8
eXDI 1-11
Exiting Multi-ICE server 3-4
Extensions debug 1-4

F

Failed to connect 5-11
Fault Address Register 5-14
Fault Status Register 5-14

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FCC notice iv
Files
  autoconfig.cfg 3-9
  configuration 3-9
  driver .mul 5-16
  log 3-4
  Non_tcp_ip.reg 2-18
Flash memory
  Memory
    flash 4-12
Flush debug handler cache if running on exit 4-17
FPGA 1-14, 4-12

H

Hard disk usage 2-3
Hardware requirements 2-3
Harvard architecture 4-40
HBI-0004 2-11
HBI-0008 2-11
HHI-0016 2-11
Hot-debug enabled 4-17
HPI-0027 1-2, 2-9, 4-47, 5-14
HPI-0068 2-12
HP-UX 1-11, 2-4

I

ICE Extension Unit. *see* IEU
IDC header F-2
IEEE 1284 parallel port
  Parallel port
    IEEE 1284 mode 3-19
IEU 1-5, 4-69
Image
  downloading 3-20
  endian 4-18
  load symbols 4-46, 4-48
Initialisation failed 5-11
Installing Multi-ICE 2-14
irlength.arm A-3

J

Joint Test Action Group. *see* JTAG
JTAG 1-2
  boundary scan 6-11
  cable length 6-15
  clock speed 3-21–3-24
    calculating 3-22
    setting 3-21
  clock too fast 5-5
  controlling settings for 3-7
  debugging support 1-4
  defining interaction 3-28
  Enabling tristate driver 5-4
  list of signals F-2
  low clock rates 3-23
  port synchronizer 6-4, 6-5
  settings 3-7
  signal list F-2
  14-way to 20-way adaptor 1-2, 4-47

K

Killing TAP connection 3-6

L

Leave processor in Halt mode on exit 4-17
Leave processor in Monitor mode on exit 4-17
Linux 1-11, 2-4
Listing TAP controllers 3-6
Little endian 4-18
LM339 G-4
Loading
  configuration files 3-3
lockdown 4-61
Log files 3-4
Logic levels 6-12
  TTL 3-19, 3-21, G-4
Logic types
  CMOS 6-14
  TTL 6-14

M

MAX823 6-7
Memory regions
  ARM740T, ARM940T 4-40
  Management Unit 5-13
Messages
  creating log files 3-4
Microsoft WinCE 4-44
Microsoft Windows
  Browser service 4-11
  browsing networks 1-13
  Desktop Update 4-3
  installing Multi-ICE 2-14
  NT 4 4-3
  parallel port driver 3-19
  requirements 2-3
  versions 2-2
  2000 1-12, 1-13
  95 4-3
Mixing ARM CPUs with others 6-3
MMU 5-13
Modification box (PCB) 2-11
Multi-ICE 3-4
  compatibility 1-3
  connection configuration 4-8
  DLL 1-10
    using on UNIX 1-7
  install directory 4-4, 4-6
  interface unit 1-7, 2-15, D-9
  multiple processors 4-27
  poll rate 3-30
  run control 3-26
  server 1-8
  server poll frequency 3-30
  startup menu 2-14
  toolbar 3-5
Multi-ICE interface unit 4-45, 5-7
Multi-ICE menus
  Connection 3-6
  File 3-2
  Help 3-8
  overview 3-2
  Run Control 3-5
  Settings 3-7
  View 3-5
Multi-ICE power supply 2-12
Multi-ICE server 2-2
  browsing for 1-13, 4-11
  configuration 3-9
  connecting to 5-11
  error messages 5-12
  multiple connections 1-8

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

starting 2-15
Multi-ICE.dll 4-4, 4-6
Multiple TAP controllers 6-9

N

Network
dialup 2-3
software 2-3
Network connections
allowing 2-17, 4-12
nICERST. *see* Signals
Non_tcp_ip.reg 2-18
Notices
CE conformity iv
FCC iv
nSRST. *see* Signals
nSYSRST. *see* Signals
nTRST. *see* Signals

O

OEM licenses 1-3
Open collector output F-3

P

Parallel port
driver 1-7
ECP mode 3-19
EPP mode 3-19
selecting 3-7
settings 3-7, 3-18
4-bit data transfer 3-19
4-bit mode 3-19
8-bit mode 3-19
PCB connections 6-12
Persistence
debugger settings 4-26
PID7T 4-64
Platform Builder 1-11
Polling devices 3-30
Portmap service 1-9, 3-17
Power
resetting 4-62
series resistor 2-12
supply 2-9, 4-45
supply requirements 2-12
Vdd 2-12, F-3
Processors
ARM10 4-65
ARM10 family 4-39–4-40
ARM1020T 1-13, 4-39, D-2, E-24–E-27
ARM10200T 1-13, 4-39, E-24–E-27
ARM7 family 4-37
ARM7DI 4-37
ARM7DMI 4-37
ARM7EJ-S 1-11
ARM7TDI 4-69
ARM7TDI-S 1-13, 4-37
ARM7TDMI 1-11, 2-17, 6-3
ARM7TDMI-S 1-11, 1-13, 4-37, 6-3
ARM710T 4-37, E-2, E-3
ARM720T 4-37, 4-44, E-4
ARM740T 4-37, E-5
ARM9 family 1-15, 4-38, 4-40
ARM9E-S 1-13
ARM9TDMI 4-38, 4-69
ARM9TDMI-S 4-38
ARM920T 1-13, 1-14, 4-38, 4-44, 6-3, E-6–E-9
ARM922T 1-11, 1-12, E-6–E-9
ARM925T 1-11, 1-12, E-10–E-13
ARM926EJ-S 1-11, E-14–E-17
ARM940T 1-14, 4-38, E-18–E-20
ARM946E-S 1-11, 1-12, 1-13, 4-40, E-21–E-23
ARM966E-S 1-11, 1-13, 1-14, 4-38, 4-40
Atmel AT91 5-9
description files 1-12
Harvard architecture 4-40
Intel XScale family 1-12
list of supported 2-5, 5-3
Samsung KS32C50100 1-13, 4-37, 4-41
Samsung S3C4510B 1-11, 4-37, 4-41
XScale microarchitecture 1-11, 1-12, 1-13, 4-39, 4-65, 5-8, D-3, E-28
Program counter 4-56

R

RDI 1-2, 1-15
version setting 4-19
Read-ahead Cache 4-19
RealMonitor, ARM 1-14, 4-21
Remote Debug Interface. *see* RDI
Report non-fatal errors 4-20
Requirements
hardware 2-3
software 2-2
Reset
circuit 6-8
EmbeddedICE logic 4-63
power-on 4-62
signals 6-6
simulating 4-62
system 4-42
vector 4-64
ROM 4-56
at address zero 4-58, 4-64, 5-8
breakpoints in 4-62
debugging applications 4-62
RPC
display 3-5
logging 3-4
RS422 6-15
Running, Multi-ICE server 2-15

S

S bit in vector_catch 4-50, B-2, B-3
Scan chains 1-4
displaying 3-10
examining 3-3
requirements 2-5
Self-modifying code
debugging 4-56
Semihosting 4-50
breakpoint usage 4-53
Enabling and disabling 4-50
Server
configuring 2-16, 3-9
starting 2-15
Settings
clock speed 3-7
debuggers
saving 4-26

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JTAG clock speed 3-21
parallel port 3-7, 3-18
RDI version 4-19
user output bits 3-19, 3-21, G-4
Signals
BREAKPT 1-4
DBGACK 1-4, 4-58
DBGEN 5-5, 5-6
DBGRQ 1-4, 3-27, 4-58, 5-5
drivers, ASIC 6-15
JTAG F-3
JTAG, list of F-2
MCLK 5-4
multiplexing JTAG 6-10
nICERST Glossary-4
nSRST 3-4, 3-5, 4-58, 5-3, 6-7, 6-13, 6-14, A-4, D-5
nSYSRST Glossary-4
nTDOen 5-4
nTRST 3-4, 3-5, 4-58, 5-3, 5-6, 6-3, 6-6, 6-13, A-4
nWait 4-59
open collector F-3
pull-up resistors 4-46, 4-47
reset 6-6
RTCK 3-24, 5-8, 6-13, 6-17
RTCK feedback loop 6-4
TCK 3-22, 6-13, 6-15, A-4
frequency F-7, F-11
14-way adaptor 6-17
TDI 6-13
Timing F-5
TMS 6-13
Vsupply 2-12, 4-46, 6-17
VTref 4-46, 4-47, 6-13
Single-stepping breakpoints B-3
Software requirements 2-2
Solaris 1-11, 1-13, 2-4
Standard EmbeddedICE Interface Unit connection 2-9
Standards
IEEE 1149.1 1-2, 2-4, 3-16
Multi-ICE connection 2-9
RS422 6-15
Starting
CPU cores 3-5
Multi-ICE Server 2-14
Stopping
CPU cores 3-5
Supported processor list 2-5, 5-3
SWI vector
breakpoint B-2
handlers 4-42, 4-50
handlers, adding 4-52
Symbols
core status, JTAG 3-10
image load 4-46–4-48
Synchronizing
clocks 3-24
processor start and stop 1-8, 3-5, 3-25–3-27, D-2
Synchronous
start group 3-26
stopping processors 3-26
System requirements 2-2

T

T bit in CPSR B-4
TAP controllers 1-5, 6-9
bypass 6-3
chaining 6-3
IR register A-2
listing 3-6
multiple 6-9
user output bits 3-20
TAPOp
controlling user output bits 3-21, G-4
protocol version 3-14
user output bits 3-19
TAPOp procedure calls
TAPOp_WriteMICEUser1 3-20
TAPOp_WriteMICEUser2 3-20
Target
debugging multiple 4-27
interface voltage levels 6-13
TCK 6-13, A-4
Test Access Port 1-5
controllers 6-2
Texas Instruments JTAG adaptor 2-12
Thumb state flag B-4
ThumbCV 4-24, 4-25
Timeout
hardware interface 5-7
Toolbars 3-5
Trace Debug Tools 1-13, 4-22
trace capture dll 4-23

U

UNIX
connecting debugger 4-3
debugging 2-2
support 1-3
Unknown
device 2-16, 5-3
watchpoint error 4-53
User input bits
sample circuit 3-21, G-4
viewing 3-21, G-5
User I/O connector G-2
User output bits
accessing G-4
controlling 3-7
setting 3-19, 3-21, G-4
setting TAP controllers 3-20
TAPOp procedure calls 3-19, 3-21, G-4

V

V bit in CP15 4-44
Variable
vector_catch 4-58
Variables
arm9_restart_address 1-14
cache clean code address 5-12, 5-13
cp15_cache_selected 4-38, 4-39, 4-40, E-2
cp15_current_memory_area 4-37, 4-40, E-2
cp_access_code_address 4-37–4-39
debugger internal 4-36
icebreaker_lockedpoints 4-37–4-41, 4-69, B-5
internal_cache_enabled 4-19, 4-37–4-41
internal_cache_flush 4-19, 4-37–4-41
ks32c_special_base_address 4-37, 4-41
safe_non_vector_address 4-37–4-41

Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

searchpath 4-36
semihosting_dcchandler_address 4-36–4-42, 4-51, 4-52, 5-9
semihosting_enabled 3-27, 4-24, 4-36–4-39, 4-42, 4-50–4-52, 4-58
semihosting_vector 4-50, 4-52, 4-53, 4-58
sw_breakpoints_preferred 4-37–4-39, 4-42
system_reset 4-37–4-39, 4-42
top_of_memory 4-37–4-39, 4-42, 4-63, 5-9
user_input_bit1 4-37, 4-38, 4-39, 4-43
user_output_bit1 4-37, 4-38, 4-39, 4-43
vector_address 4-36, 4-37, 4-38, 4-39, 4-41, 4-44, 4-50
vector_catch 4-36, 4-57, 4-64
Vector catch breakpoints B-2
Vector catch hardware 4-58
ARM10 processor B-2
ARM9 processor B-2
XScale processor B-2
Viewing
device information 3-15
user input bits 3-21, G-5

W

Warranty 1-2
Watchpoints 4-55
unknown 4-53
Windows. *see* Microsoft Windows

X

XScale processor
debug architecture D-3
debug handler 4-16
reset vector overloading D-5

Numerics

4-bit data transfer 3-19
4-bit parallel port 3-19
8-bit parallel port 3-19

 Copyright © 1998-2002 ARM Limited. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP