ARM® Agilent® Debug Interface

Version 1.0

User Guide

ARM

ARM DUI 0158A

ARM Agilent Debug Interface
User Guide

Release Information

The following changes have been made to this document.

Change history

Date	Issue	Change
September 2001	A	Release 1.0.

Proprietary Notice

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Contents
ARM Agilent Debug Interface (ADI) User Guide

Preface

This preface introduces the ARM Agilent Debug Interface Version 1.0 User Guide. It explains the structure of the user guide and lists other sources of information that relate to the Agilent Gateway interface unit and to ARM debuggers.

This preface contains the following sections:

About this document

This document describes the ARM Agilent Debug Interface Version 1.0 User Guide.

Intended audience

This document is written for users of the Agilent Emulation Probes who are using the *ARM Developer Suite* (ADS) development environment. It is assumed that you are a software engineer with some experience of the ARM architecture.

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 ARM ADI product
- what has changed between this version and the last.

Chapter 2 ***Configuring ARM ADI***

This chapter explains ARM ADI configuration in detail, including the configuration for multiprocessor targets and execution trace capture.

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.

`typewriter` Denotes text that can be entered at the keyboard, such as commands, file names and program names, and source code.

`type`writer 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.

Further reading

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

ARM publications

This document contains information that is specific to ARM ADI:

- *ARM ADI Installation Guide* (ARM DSI 00018).

If you are using ARM ADI with the *ARM Developer Suite* (ADS) v1.1, refer to the following books in the ARM *Trace Debug Tools* (TDT) product and in the ADS document suite for more information:

- *Trace Debug Tools Version 1.1 User Guide* (ARM DUI 0118)
- *Getting Started* (ARM DUI 0064)
- *CodeWarrior IDE Guide* (ARM DUI 0065)
- *ADS Debuggers Guide* (ARM DUI 0066)
- *ADS Developer Guide* (ARM DUI 0056).

The following ARM documents are also available:

- *ARM Application Library Programmers Guide* (ARM DUI 0081)
- *ARM7DI Datasheet* (ARM DDI 0027)
- *ARM710T Datasheet* (ARM DDI 0086)
- *ARM720T Datasheet* (ARM DDI 0087)
- *ARM740T Datasheet* (ARM DDI 0008)
- *ARM7TDMI (Rev 3) Technical Reference Manual* (ARM DDI0029)
- *ARM9TDMI (Rev 3) Technical Reference Manual* (ARM DDI 0180)
- *ARM920T (Rev 1) Technical Reference Manual* (ARM DDI 0151)
- *ARM922T (Rev 0) Technical Reference Manual* (ARM DDI 0184)
- *ARM940T (Rev 2) Technical Reference Manual* (ARM DDI 0144)
- *ARM946E-S (Rev 1) Technical Reference Manual* (ARM DDI 0201)
- *ARM966E-S (Rev 1) Technical Reference Manual* (ARM DDI 0186).

The following additional documentation that might be useful, and 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 in PDF format.

Other publications

This section lists relevant documents published by third parties:

- IEEE Std. 1149.1-1990, *Standard Test Access Port and Boundary-Scan Architecture*.

Refer to the following publications for additional information about the Agilent analyzers described in this manual:

- E5903-97002, *Trace Port Analysis for ARM ETM User's Guide*, Agilent Technologies, 2000.
- E3459-97005, *Emulation for the ARM7/ARM9 User's Guide*, Agilent Technologies, 2000.
- E5904-97000, *Agilent Technologies E5904B Option 300 Trace Port Analyzer for ARM User's Guide*, Agilent Technologies, 2000.

To access these documents, see the website `http://www.agilent.com`.

Feedback

ARM Limited welcomes feedback both on ARM ADI and on the documentation.

Feedback on ARM ADI

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

- the ARM ADI 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.

Chapter 1
Introduction

This chapter introduces the *ARM Agilent Debug Interface* (ADI). It is split into the following sections:

1.1 About ARM ADI

ARM ADI is a software product that enables a *Remote Debug Interface* (RDI) 1.5.1 compliant debugger to use an Agilent Emulation Probe to debug software running on ARM processors. ARM ADI is designed to run with the *ARM eXtended Debugger* (AXD).

ARM ADI uses an Ethernet network to communicate between the workstation running the ARM ADI software and the Agilent hardware. Agilent hardware that is compatible with ARM ADI uses the *Standard Test Access Port and Boundary-Scan Architecture* interface defined by the *Joint Test Action Group* (JTAG) to communicate with the processor.

The ARM ADI product enables you to:

- load and run programs on one or more connected processors
- examine the state of the target processor registers and memory
- collect trace information from an *Embedded Trace Macrocell* (ETM) connected to the processor, when used in conjunction with ARM *Trace Debug Tools* (TDT) and suitable Agilent trace capture hardware.

The ARM ADI product contains software and documentation that enable an ARM debugger to communicate with the Agilent interface unit. The product includes the following components:

- an Install Guide in PDF format
- two *Dynamic Link Library* (DLL) files to use with the debugger
- configuration files for each of the supported ARM processors
- a User Guide in PDF and Dynatext formats.

ARM ADI is only available from the ARM Limited website, `www.arm.com`. It does not include a debugger. Suitable debuggers include:

- ARM AXD, supplied with *ARM Developer Suite* (ADS) Version 1.1 or better
- ARM ADW, supplied with ADS Version 1.1, excluding support for `Gateway2.dll`
- third-party debuggers that conform to the ARM RDI 1.5.1 interface.

ARM ADI supersedes the Gateway software that is included in SDT Version 2.51 and in ADS up to Version 1.1.

1.2 The components of ARM ADI

ARM ADI contains two distinct software components, called Gateway and Gateway2, that supersede the components included in ADS 1.1. These components perform the following functions:

Gateway DLL Drives Agilent emulation hardware (such as the Emulation Probe or the Emulation Module) to provide access to target system memory and run control for one or more processors. It is similar in function to ARM Multi-ICE®.

Gateway2 DLL Drives Agilent trace data capture hardware. This hardware reads from the data port of an Embedded Trace Macrocell, enabling a debugger to display a history of instructions executed by a processor without impacting on system performance or behavior. Gateway2 requires TDT.

The relationships between the target, the Agilent hardware, ARM ADI, and the debugger are shown in Figure 1-1.

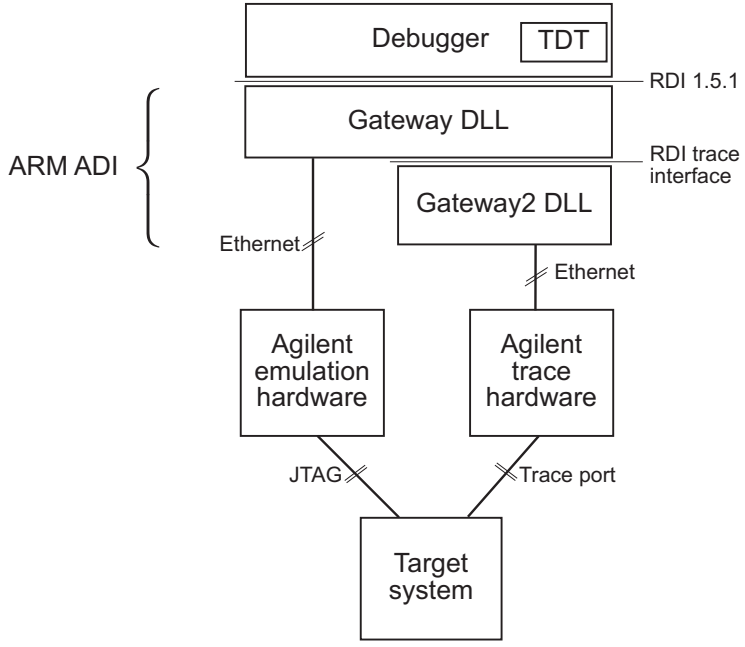

Figure 1-1 How the Agilent hardware, the ARM ADI software, and the debugger are combined

1.3 Hardware and software requirements

The ARM ADI product requirements are described in the following sections:

- *Workstation hardware requirements*
- *Workstation software requirements*
- *Agilent hardware requirements for Gateway*
- *Agilent hardware requirements for Gateway2* on page 1-5.

1.3.1 Workstation hardware requirements

The ARM ADI software runs on IBM-PC compatible workstations with the following characteristics:

- an x86 Pentium-class processor or better
- 32Mb memory
- 8Mb free disk space
- a 10 Base-T or 100 Base-T Ethernet network interface.
- an SVGA monitor or better

1.3.2 Workstation software requirements

The ARM ADI software runs on a workstation with the following characteristics:

- Microsoft Windows 95, Windows 98, Windows Me, Windows NT 4.0, or Windows 2000
- an installation of one of the following:
  - — ADS Version 1.1 or better
  - — a third party debugger that conforms to RDI 1.5.1.
- for Gateway2 execution trace function, an installation of TDT Version 1.1 or better.

1.3.3 Agilent hardware requirements for Gateway

The ARM ADI software requires one of the following items of Agilent hardware:

- an E5900B emulation probe
- an E5904B emulation probe
- a 16600 or 16700 series Logic Analyzer and E5900 Emulation Module.

1.3.4 Agilent hardware requirements for Gateway2

The Gateway2 software requires one or more of the following items of Agilent hardware in addition to those required by Gateway (detailed in *Agilent hardware requirements for Gateway* on page 1-4):

- an E5904B Emulation Probe (includes trace buffer)
- a 16600 or 16700 series Logic Analyzer.

1.4 Basic principles

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

- *Debug extensions to the ARM processor*
- *The EmbeddedICE logic*
- *How ARM ADI differs from a debug monitor* on page 1-7.

Note

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

1.4.1 Debug extensions to the ARM processor

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

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

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

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

ARM ADI uses these, and other signals, by using the debug interface of the processor, 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 processor. A list of sources is provided in *Further reading* on page vii.

1.4.2 The EmbeddedICE logic

The EmbeddedICE logic is the integrated on-chip logic that provides JTAG debug support for ARM processors. EmbeddedICE/RT is a superset of EmbeddedICE that includes extensions supporting real-time debug, including setting breakpoints on a running target. The EmbeddedICE logic is accessed through the TAP controller on the ARM processor using the JTAG interface.

The standard EmbeddedICE logic consists of:

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

For more details on using the DCC in your software, see the *ADS Developer Guide*.

You can program one or both of the watchpoint units to halt the execution of instructions by the ARM processor. 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.

1.4.3 How ARM ADI 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 some resource (for example, memory, access to exception vectors, and time).

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 processor, that is, parts that enable the host to communicate with the target
- an external interface unit that buffers and translates processor 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.

1.5 Changes from the ADS Gateway software

This section describes the changes and new features that have been added to the product since the release of the Gateway software in ADS Version 1.1:

- *New features in ARM ADI*
- *Changes in ARM ADI*.

1.5.1 New features in ARM ADI

The new features in ARM ADI are:

New processor support

ARM9 processors ARM946E-S™ (Rev 0).
ARM922T™ (Rev 0).

Support for multiprocessor operation

ARM ADI supports connections to one of several processors in a multiprocessor target. Using two or more debuggers, you can also debug or trace more than one processor at a time.

New operating system support

Windows Me is now a fully supported operating system.

Better Agilent probe support

Support for the E5904B Agilent Emulation Probes in ARM ADI has been improved.

New documentation

There are new online help files and a User Guide.

1.5.2 Changes in ARM ADI

Other changes in ARM ADI include:

Improved User Interface

The configuration dialogs have been improved to make configuration easier and more flexible.

E5903 Agilent probe support removed

The E5903 Agilent Emulation Probe is not supported by ARM ADI.

Chapter 2
Configuring ARM ADI

This chapter describes how to set up ARM ADI to connect a debugger to one or more processors. It contains the following sections:

2.1 Setting up and using ARM ADI

This section explains how to configure the debugger and the ARM ADI software.

Note

- These instructions assume that you have already set up your Agilent Gateway interface unit. Refer to the manuals for your hardware for more information.
- If you are using the Agilent E5900B and the probe is connected to the target, power on the probe and wait until it is completely initialized before powering the target. To use the probe on a running target, ensure the probe is not connected to the target and follow the procedure in *Hot-plugging the Agilent Emulation Probe* on page 2-24.

The procedure to configure the ARM ADI software is shown in Figure 2-1.

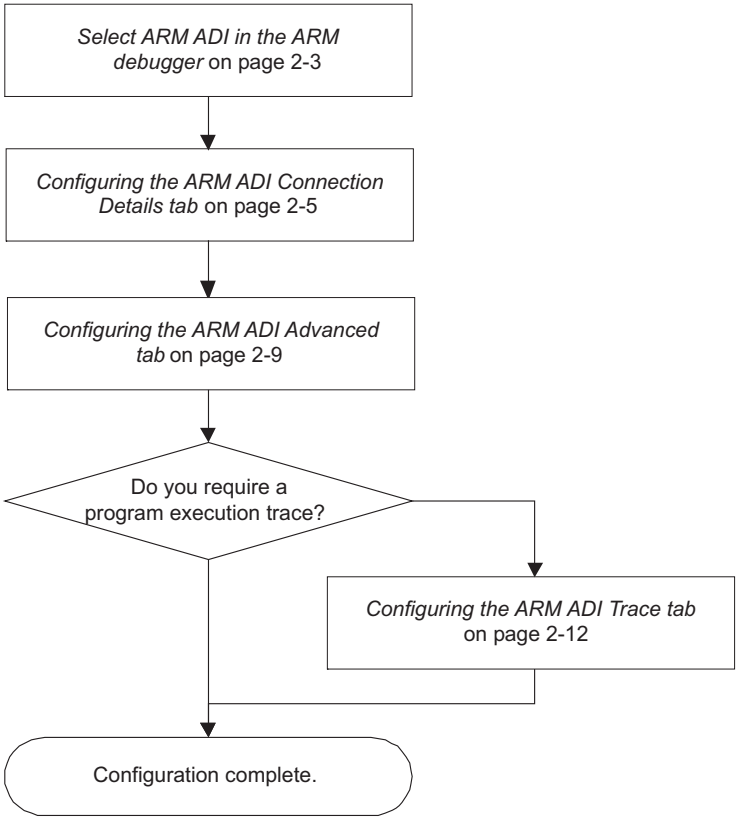

Figure 2-1 Configuration procedure

2.1.1 Selecting ARM ADI in the ARM debugger

The procedure used to load the ARM ADI software to the debugger varies between different debuggers. This section assumes that you are using the ARM AXD debugger. Refer to the documentation for your debugger for more information about configuring RDI component DLLs.

The procedure is described in the following sections:

1. *Configuring Microsoft Windows to display DLL files*
2. *Configuring the debugger*.

Configuring Microsoft Windows to display DLL files

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

To show `.dll` files in 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.

To show `.dll` files in Windows 95 and 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.

Configuring the debugger

This section describes how to connect the ARM ADI DLL to the ARM AXD debugger.

1. Using the **Start → Programs** menu run an instance of AXD.
2. Select **Options → Configure Target** (Figure 2-2 on page 2-4).

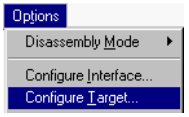

Figure 2-2 The AXD Options menu

This displays the **Choose Target** dialog shown in Figure 2-3.

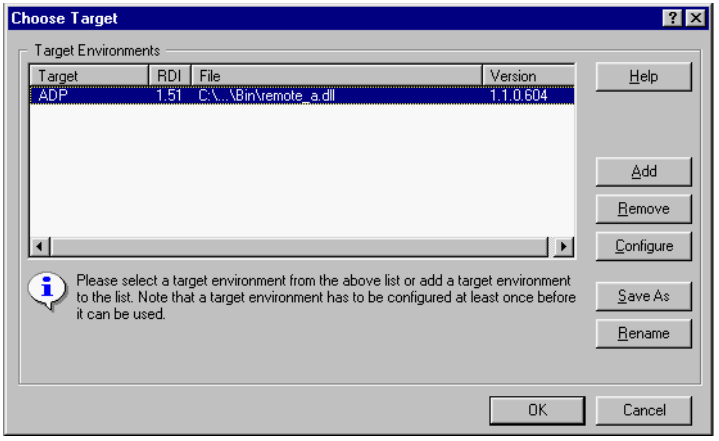

Figure 2-3 The AXD Choose Target dialog

3. If Gateway is listed in the **Target Environments** list, select it (by clicking on it) and go on to step 4. If it is not listed:
   a. Select **Add**. A Windows **Open** dialog is displayed.
   b. Navigate to the ARM ADI install directory (for example, `C:\Program Files\ARM\ADS_1.1`).
   c. Find the file `Gateway.dll`, select it, and click on the dialog **Open** button as shown in Figure 2-4 on page 2-5.

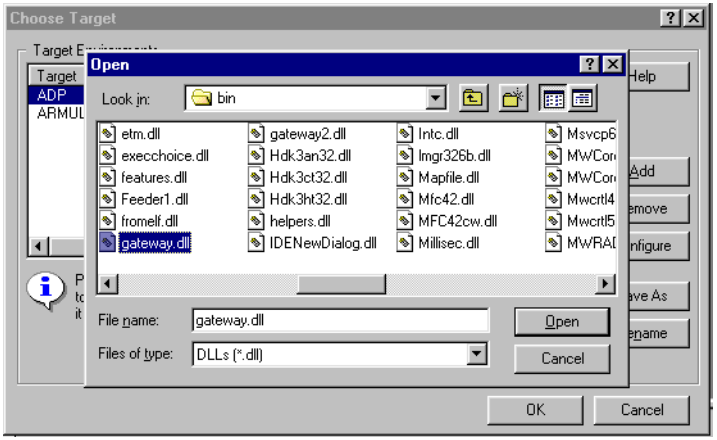

Figure 2-4 Selecting the Gateway DLL using AXD

Clicking on **Open** dismisses the dialog.

The file path name of the Gateway DLL is displayed in the **Target Environments** list.

Note

You can ensure that AXD starts with Gateway in the list by configuring a single connection with Gateway, then quitting and restarting AXD.

4. Select **Configure** to display the ARM ADI configuration dialog.

2.1.2 Configuring the ARM ADI Connection Details tab

The Connection Details tab displayed in Figure 2-5 on page 2-6 is used to specify the information the ARM ADI software requires to access the Agilent Gateway interface unit.

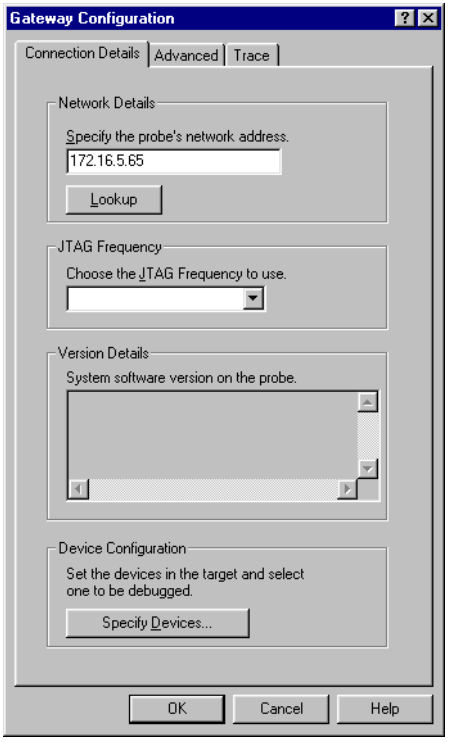

Figure 2-5 Gateway Configuration, Connection Details tab

You configure the dialog as follows:

1. Enter in the **Network Details** text field the name or network address of the Agilent Gateway interface unit, as:
   - a name, if there is an entry in your `hosts` file or in the *Domain Name Service* (DNS) database for your domain.
   - an *Internet Protocol* (IP) address, in dotted-quad format.

   If you do not know the IP address of the Agilent Gateway interface unit, refer to the Agilent documentation for details of how to view or set it.

2. If necessary, click on the **Lookup** button. The lookup is done automatically when keyboard focus leaves the **Network Details** field and the text in the field has changed. However, to retry connecting to the same address you must use **Lookup**. If you are unsure what to do, click on **Lookup**.

3. Use the **JTAG Frequency** control to select the frequency at which the Agilent Gateway interface unit clocks data across the target JTAG port. You must select one of the listed frequencies before the Agilent Gateway interface unit can connect to the target.

   Note

   The Agilent Gateway interface unit probably supports JTAG frequencies that are higher than your target hardware supports. Using these higher frequencies results in the Agilent Gateway interface unit failing to connect, or connecting unreliably.

   If the target hardware supports it, you can select Adaptive Clocking. This means that the outgoing clock (**TCK**) changes as fast as the target can acknowledge the previous clock tick (using **RTCK**). Therefore the clock runs at almost the maximum speed of the target, but logic delays prevent it running at full speed.

   Many ARM processors have a maximum JTAG frequency of 10MHz. Several synthesizable ARM processor designs require slower frequencies, for example 5MHz, or the use of Adaptive Clocking, for reliable operation.

   When there are multiple joins in the JTAG cable, or the JTAG cable is more than about 10cm (4 inches) long, ARM recommends that you start debugging with a JTAG frequency of 1MHz or less. If debugging at this frequency is reliable you might try using Adaptive Clocking, or increase the frequency to 5MHz or higher.

4. Click on the **Specify Devices** button to display the Specify Devices dialog displayed in Figure 2-6 on page 2-8. If you:

   - have configured this Agilent Gateway interface unit in a previous instance of the debugger and the session was saved, the configured processor list is read from the last session and displayed in Current Devices.

     Note

     Do not change the configured device list with another debugger connected.

   - have not configured the processor list in another debugger, the Current Devices list is blank. You must fill it in completely before you can connect to any of the processors.

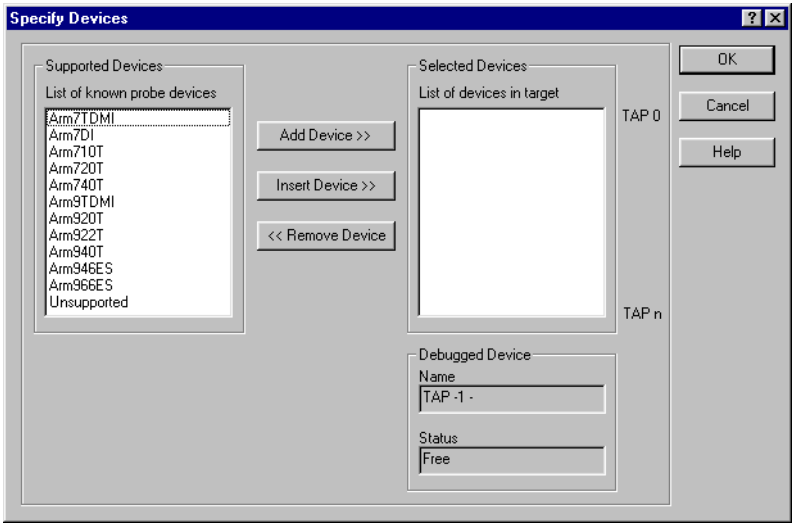

Figure 2-6 Gateway Configuration, Specify Devices dialog

5. Specify all of the connected processors in order. For each processor in the JTAG chain, starting from the processor that is nearest **TDO** on the Agilent Gateway interface unit:

   a. Select the processor type in the Supported Devices list.

   b. Click **Add Device>>** to add it to the Current Devices list.

   If you wish to change the order of devices in the Current Devices list, you can use the **Insert Device>>** button to add a device at a particular point in the Current Devices list. To do this:

   a. Select in the Current Devices list a device that is to follow the new device.

   b. Select the name of the new device in the Supported Devices list.

   c. Click **Insert Device>>**.

   You can remove a device from the Current Devices list by:

   a. Select the device to be removed.

   b. Click **<<Remove Device**.

   If there is any TAP controller on your target that does not appear in the list of supported devices, select Unsupported. Clicking **Add Device>>** then displays the Unsupported Device dialog, shown in Figure 2-7 on page 2-9:

   a. Enter the number of bits in the device TAP controller instruction register. This information enables the Agilent Gateway interface unit to bypass this TAP controller.

   b. Click **OK**.

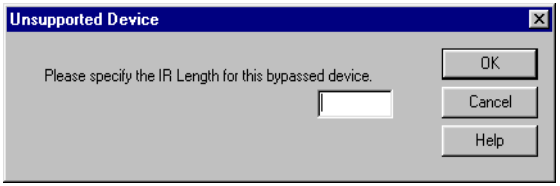

Figure 2-7 The Unsupported device dialog

6. Select the processor that you want to debug in the Current Devices list. The status of the selected processor is shown in the Debugged Device Name and Status fields.

   The Status field indicates whether you can connect to the device that is selected in the Current Devices list. If the status is:

   Free You can connect to the device.

   In Use You cannot connect to the device because another debugger is already using it.

   In Use (connected) You are already connected to the device. You can still modify the connection details if required.

7. Click **OK** to close the Specify Devices dialog.

2.1.3 Configuring the ARM ADI Advanced tab

The Advanced tab displayed in Figure 2-8 on page 2-10 contains additional information about the target.

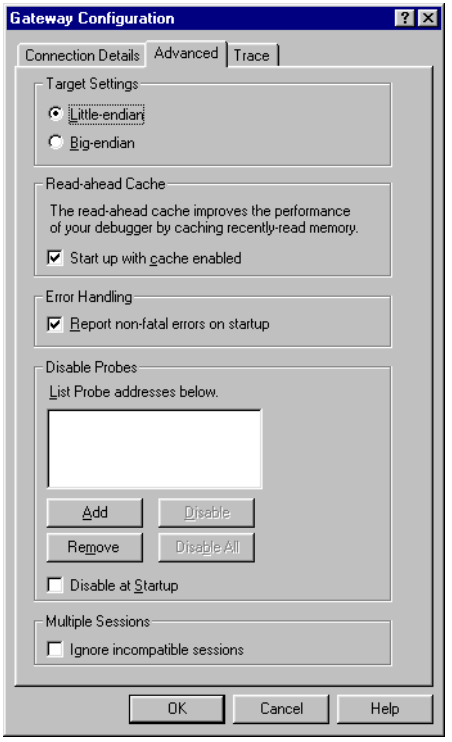

Figure 2-8 Gateway Configuration, Advanced tab

You configure the dialog as follows:

1. If your target is little-endian, so the least significant byte of each word is at the lowest memory address, select **Little-endian**.
2. If your target is big-endian, so the most significant byte of each word is at the lowest memory address, select **Big-endian**.
3. Uncheck the **Start-up with cache enabled** checkbox if your target:
   - uses DMA to memory that you want to read or set using the debugger
   - dynamically changes the MMU page translations in the memory region that you are debugging
   - uses a bootup-memory map that is different from the normal memory map and you are debugging the boot sequence
   - has read-sensitive devices on the memory bus that you are using the debugger memory windows to view.

The **Start-up with cache enabled** checkbox controls the initial state of the read-ahead caching option. Read-ahead caching improves memory read performance by reading more memory than requested by the debugger. The additional memory is saved in case it is required later.

You can flush the cache at a specific time by setting the debugger variable `internal_cache_flush` to `1`. You can switch the cache off temporarily by setting the debugger variable `internal_cache_enabled` to `0`.

4. Uncheck **Report non-fatal errors on startup** if you find that an error occurs when the debugger connects to your target that causes the connection attempt to abort, but you know that you can work around the error.

   If this option is checked, then both fatal and non-fatal errors are reported and both cause the connection attempt to abort. This is the default, and means that you are informed of any problems that ARM ADI detects.

   If this option is unchecked, then the ARM ADI software ignores non-fatal errors, so that debuggers that consider that any error detected during configuration is fatal do not prevent you from connecting to your target.

5. If you are using several Agilent Emulation Probe interfaces connected to the same target in order to trace multiple processors, only one of them can control the JTAG lines at once. Enter into the Disable Probes list the network name or address of every probe that is only being used to capture trace information from your target. For each probe:

   a. Click on **Add**. The Add Address dialog is displayed (Figure 2-9).

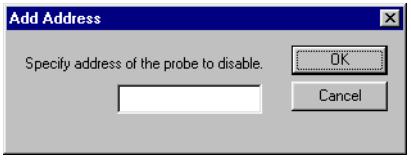

Figure 2-9 The Disable Probes, Add Address dialog

   b. Enter the network name or address of an Agilent probe into the text field.

   c. Click **OK**.

When you have entered all of the addresses, clicking **Disable All** causes ARM ADI to immediately contact the Agilent interface units and disable JTAG control.

Alternatively, you can select a particular address in the Disable Probes list and click **Disable** to immediately disable JTAG control on that particular interface unit. You must disable JTAG control on all but one of the devices that are connected to the same JTAG chain.

To remove a probe from the list, select an address in the Disable Probes list and click **Remove**.

6. If you have configured a Disable Probes list, it is saved for future debugger sessions. Click **Disable at Startup** if you want the debugger to disable the JTAG on all of the devices listed in the Disable Probes list as soon as ARM ADI starts. Because the only way to enable JTAG with ARM ADI is to power-cycle the Agilent Emulation Probe, you must be careful in selecting the devices you disable.

7. If you know that:
   - your Agilent probe has been configured incorrectly for the target it is now connected to by previous debugger sessions that have not terminated those sessions, and
   - there is no debugger currently connected to the Agilent probe

   then you must check the **Ignore incompatible sessions** checkbox.

   The default, unchecked, state of this control causes ARM ADI to abort a connection if it finds it is connecting to an Agilent probe that has another connection to it and the probe is configured differently to the debugger session information. Checking the control results in the connection being made and the new target configuration being downloaded, even if it appears that another session has a conflicting configuration. If there is another debugger still connected, you must not continue to use it.

   An alternative solution is to switch off the probe and then switch it on again.

8. Click **OK** to return to the AXD Choose Target dialog.

9. Click **OK** to connect to the target.

2.1.4 Configuring ARM ADI trace support

The Trace tab of the Gateway Configuration dialog is shown in Figure 2-10 on page 2-13. You use this dialog to locate and configure trace capture software. The trace capture software that is supplied in ARM ADI is called Gateway2 (`Gateway2.dll`).

Note

If TDT is not installed, the Trace tab is not shown and you cannot use `Gateway2.dll`.

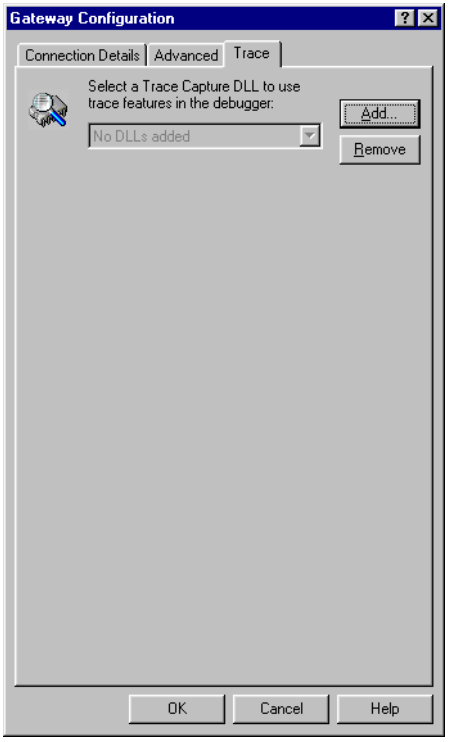

Figure 2-10 The Trace tab

You fill in the dialog as follows:

1. If the Select a Trace Capture DLL drop-down list contains an entry for `Gateway2.dll`, select it and go to step 5.
2. Click on the **Add...** button. This displays the Select Trace Capture DLL dialog as shown in Figure 2-11 on page 2-14.

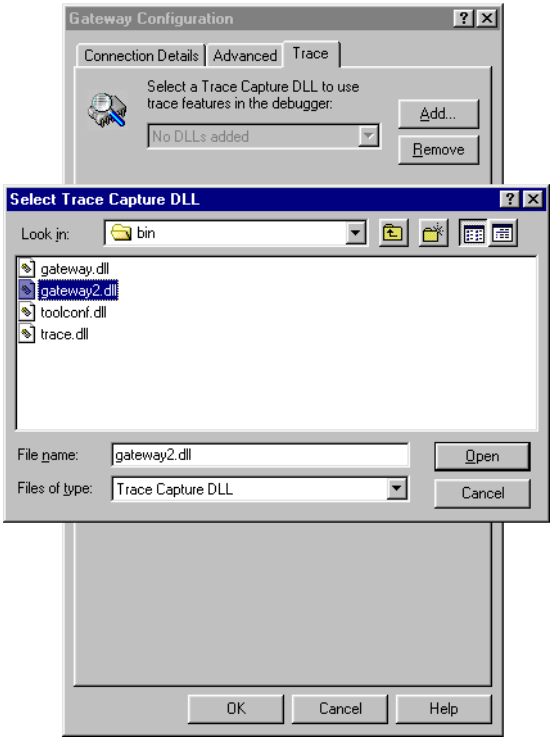

Figure 2-11 Selecting Gateway2

3. Use the controls on the Select Trace Capture DLL dialog to locate the `Gateway2.dll` trace capture DLL.

4. Click **Open**. The trace capture DLL you select is added to the Select a Trace Capture DLL drop-down list and made the current selection.

   If you select a DLL that is not a trace capture DLL, an error message is displayed. Click **OK** and return to step 2 to select another DLL.

5. Selecting `Gateway2.dll` changes the Gateway Configuration dialog as shown in Figure 2-12 on page 2-15.

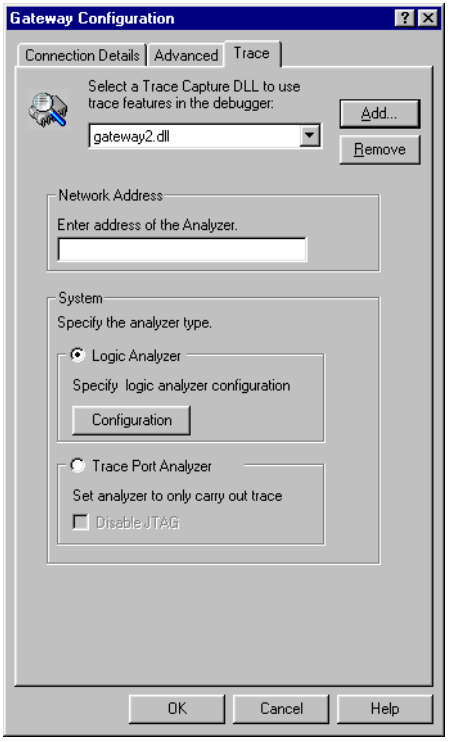

Figure 2-12 The Gateway2 controls within the Trace tab

6. Type the network name or address of the trace capture interface unit into the **Network Address** text field.

7. You must choose the next step based on the hardware you are using, as shown in Figure 2-13 on page 2-16.

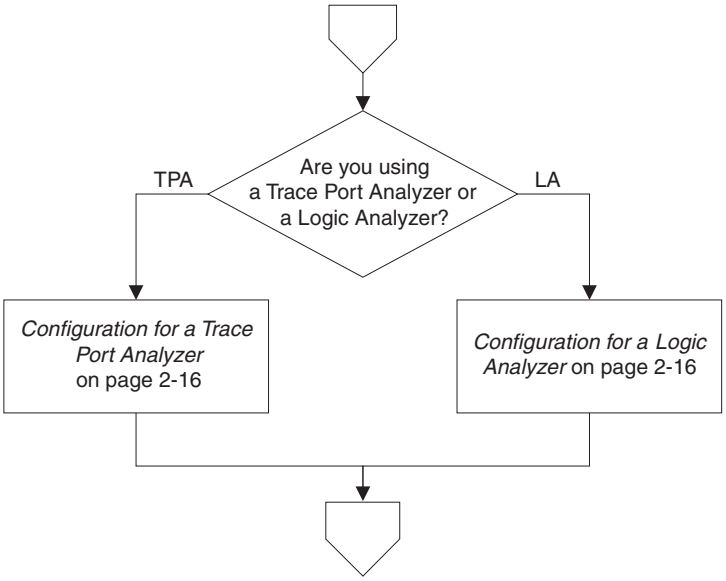

Figure 2-13 Hardware configuration procedures

Configuration for a Trace Port Analyzer

Follow these steps to set up ARM ADI for use with an Agilent Trace Port Analyzer, such as the E5904B:

1. Click **Trace Port Analyzer**.
2. Select **Disable JTAG** if the trace port analyzer is collecting only trace data, and another unit is controlling the processor.

   Note

   Do not select **Disable JTAG** if the address you entered into the **Network Address** field is the same as the address you entered into the Connection tab **Network Details** field.

Configuration for a Logic Analyzer

Follow these steps to set up ARM ADI for use with an Agilent 16600 or 16700 series Logic Analyzer:

1. Click **Logic Analyzer** to connect to the Logic Analyzer.

Note

You cannot correctly configure an Agilent Logic Analyzer frame to capture more than one ETM trace channel at once. Therefore, to trace two processors using a Logic Analyzer, you must also use an Agilent TPA or a second Logic Analyzer frame.

2. Click **Configuration** to display the Logic Analyzer Configuration dialog box shown in Figure 2-14.

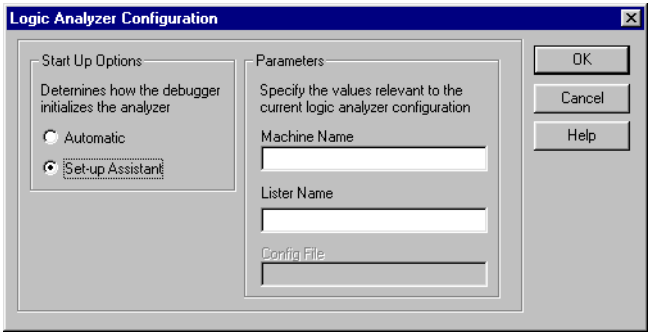

Figure 2-14 Gateway2 Logic Analyzer Configuration dialog

3. Select the appropriate startup option to indicate the level of initialization carried out by the debugger:
   - Select the **Automatic** option if you want the debugger to ensure, at start-up, that the logic analyzer is fully initialized to carry out tracing. In this case, you must specify the appropriate **Machine Name**, **Lister Name**, and **Config File** name in the dialog box.

     You must specify the full directory path to the configuration file.
   - Select the **Setup Assistant** option if you do not want the configuration file to be loaded by the debugger. You must specify the **Machine Name** and **Lister Name**. This mode is appropriate only when you are loading a configuration file from the logic analyzer user interface, which can be done using either of the following:
     — the Setup Assistant
     — the File Manager tool.

   For the default Logic Analyzer configuration files provided by Agilent:
   - the **Machine Name** is `ARM ETM Analyzer`
   - the **Lister Name** is `ETM Data`.

Note

- The default logic analyzer configuration files cannot be loaded directly by the analyzer. Instead, using the file manager tool on the analyzer user interface, you must load each file, save the configuration back using a different filename, and specify this name as the configuration to load.
- You must ensure that the Logic Analyzer configuration file setting for double edge clocking is consistent with the TDT setting for half rate clocking, as described in the *TDT Version 1.1 User Guide*.

4. Click **OK** to dismiss the Logic Analyzer Configuration dialog box and return to the Gateway Configuration dialog.

2.1.5 Configuring ARM ADI trace support for two tracing debuggers

This section describes how to set up and use two AXD debuggers to debug a target with two ETM trace connectors.

In the hardware, the JTAG chain is looped through both connectors so that you can use one TPA to control and trace either processor. However, this means that if you trace both processors, you must disable the JTAG control lines for one of the TPA units to avoid a conflict. The electrical configuration is shown in Figure 2-15 on page 2-19.

Note

You are recommended to use the procedure given in *Configuration using named AXD target configurations* on page 2-21 to save configuration information for each of the debuggers. This avoids the possibility of inadvertently overwriting the configuration of one processor with that of the other.

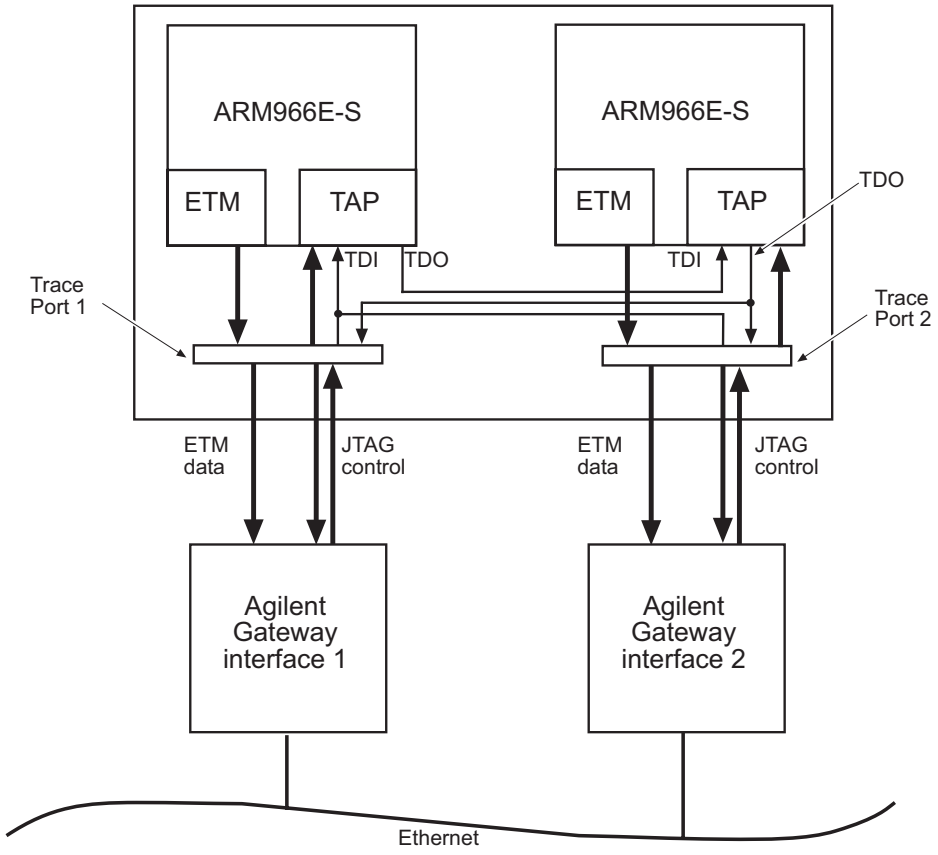

Figure 2-15 Two TPAs connected to a dual processor target

In Figure 2-15, the JTAG data lines **TDI** and **TDO** chain through the processors but both `Trace Port 1` and `Trace Port 2` are connected to the chain at the same point. This enables either trace port to be used on its own by ignoring the other port. To connect to both trace connectors, you must:

1. For the first debugger, connect both `Gateway.dll` and `Gateway2.dll` to Gateway interface 1.
2. With a second debugger, connect Gateway to Gateway interface 1 and Gateway2 to Gateway interface 2, specifying **Disable JTAG** in the Gateway2 configuration.

You configure the first debugger in the normal way, the second debugger the procedure is:

1. Start up another instance of AXD on the host workstation.

2. Click on **Configure Target**.
3. Select `Gateway.dll` in the Target Environments list and click on **Configure**.
4. Enter into the Gateway Configuration dialog:
   a. In **Network Details** the name or address of Gateway interface 1.
   b. A **JTAG Frequency** (for example, 5MHz).
5. Click **Specify Devices**. If necessary, configure the list of devices for the target so that, for example, the Current Devices list shows two `Arm966ES` devices.
6. Select the second device in the Current Devices list.
7. Click **OK**.
8. Select the **Trace** tab.
9. Select `Gateway2.dll` in the list of trace capture DLLs.
10. Enter into **Network Address** the network name or address of Gateway interface 2.
11. Click **Trace Port Analyzer**.
12. Click **Disable JTAG**, so that it is checked.
13. Click **OK**. This debugger connects to the processor and you can start using the second processor.

Figure 2-16 on page 2-21 shows two instances of AXD running, both using the same Agilent Gateway interface unit to connect to the dual processor target.

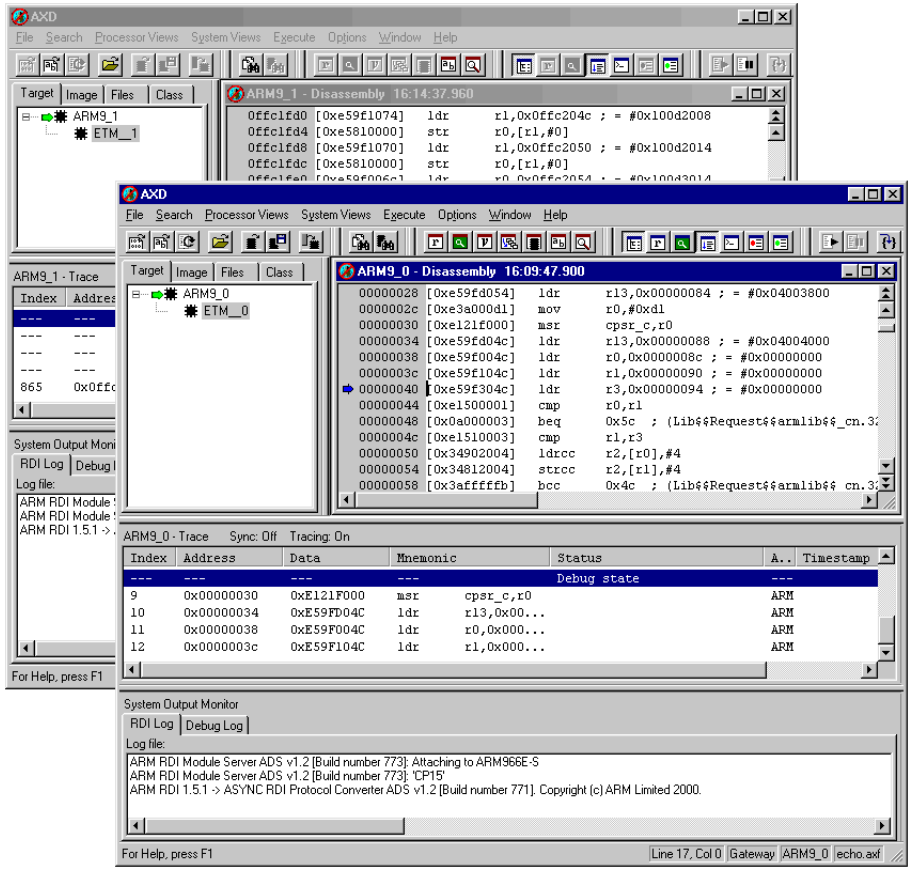

Figure 2-16 Connecting to a dual processor target

2.1.6 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 ARM ADI to the list of targets using the **Add** button if it is not already present.

4. Copy the ARM ADI target configuration to create one target configuration for each processor on the target board, as follows:
   a. Select the `Gateway` 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 2-17.
   d. Click **OK**.

   For example, in a three processor setup you can create two more copies of ARM ADI called `Analyzer1` and `Analyzer2`, and then you can click **Rename** to rename the original to `Analyzer0`.

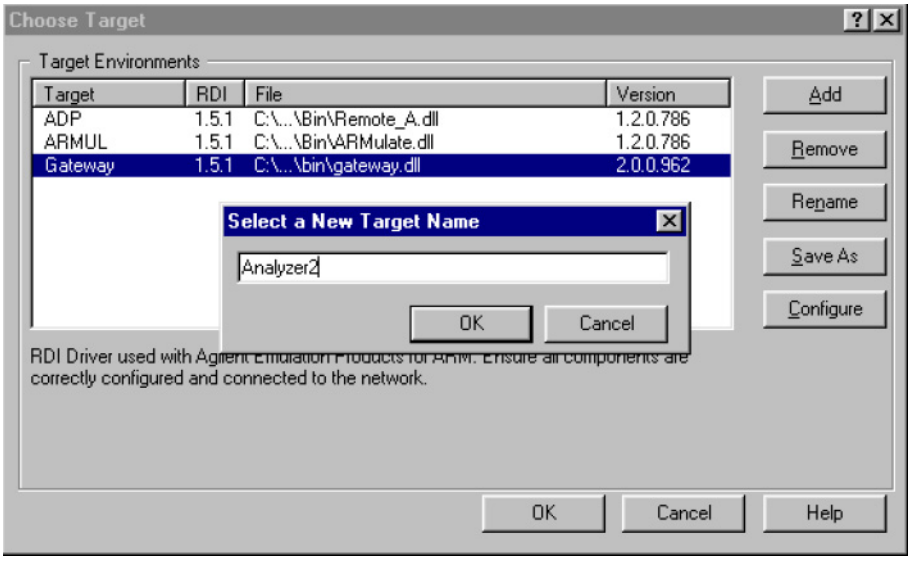

Figure 2-17 Saving a named target configuration

5. Configure these targets separately by selecting the name in the Target Environments list and clicking **Configure**.

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 that ARM ADI 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.

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*.

2.1.7 Disabling and removing the Gateway2 trace component

If the Gateway2 DLL has been added to the drop-down list, you can disable trace support by selecting `No Trace` in drop-down list, as shown in Figure 2-18.

You can enable trace support again by selecting the `Gateway2.dll` entry.

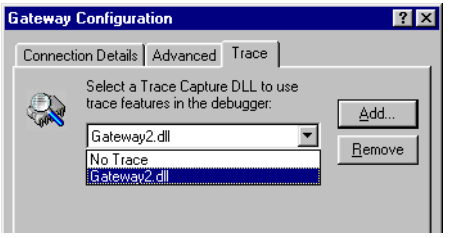

Figure 2-18 Selecting No Trace

Click on the **Remove...** button to remove the currently selected trace capture DLL from the drop-down list.

2.2 Hot-plugging the Agilent Emulation Probe

If you must connect an emulation probe to a target that is powered up and running (called hot-plugging the target), you must use the following procedure, starting with the probe connected to the network:

1. Power-on the Emulation Probe, leaving the target unconnected.
2. Configure ARM ADI to connect to the Emulation Probe as described in *Setting up and using ARM ADI* on page 2-2, and attempt a connection.

   This connection attempt fails because there is no target, but gives ARM ADI the opportunity to get the Probe into the correct state to connect to a target that is running.
3. Click **Configure** in the error message dialog. This displays the Choose Target dialog.
4. Connect the Emulation Probe to the target.
5. Click OK on the Choose Target dialog to reconnect ARM ADI to the Emulation Probe. AXD now reconnects to ARM ADI without disturbing it.

Glossary

Adaptive Clocking A technique in which a clock signal is sent out by the Agilent Gateway interface unit, which waits for the returned clock before generating the next clock pulse. The technique allows the Gateway interface unit to adapt to differing signal drive capabilities and differing cable lengths.

ADS See *ARM Developer Suite*.

Agilent Gateway interface unit One of several hardware interface units produced by Agilent that control one or more target processors using a JTAG chain. They are controlled by software running on a host workstation using Gateway protocol over an Ethernet link.

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.

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 also *Little-endian*.

CPU Central Processor Unit.

DLL *See* Dynamic Linked Library

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.

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

Gateway DLL The software component of ARM ADI that controls execution of the processor, using a connection to an Agilent Emulation Probe.

Gateway2 DLL The software component of ARM ADI that reads program execution trace information using a connection to an Agilent logic analyzer or trace port analyzer. It requires an RDI 1.5.1 run control agent, to operate such as the Gateway DLL or Multi-ICE.

Host A computer which provides data and other services to another computer. *Especially*, a computer providing debugging services to a target being debugged.

ICE *See* In-Circuit Emulator.

ICE Extension Unit A hardware extension to the EmbeddedICE logic that provides more breakpoint units.

IEEE 1149.1 The IEEE Standard which defines how TAP controllers work. Commonly (but incorrectly) referred to as JTAG.

In-Circuit Emulator A device enabling access to and modification of the signals of a circuit while that circuit is operating.

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*.

Multi-ICE Multi-processor EmbeddedICE interface. ARM registered trademark.

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.

RDI *See* Remote Debug Interface

Remote Debug Interface
Is an open ARM standard procedural interface between a debugger and the debug agent. The widest possible adoption of this standard is encouraged. RDI gives the debugger a uniform way to communicate with:

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

RTCK
Returned **TCK**. The signal which enables Adaptive Clocking.

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.

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.

TMS
An IEEE1149.1 (JTAG) port signal that controls the TAP controller.

TDI
An IEEE1149.1 (JTAG) port signal on the input of the TAP data scan chain (shift register).

TDO
An IEEE1149.1 (JTAG) port signal on the output of the TAP data scan chain (shift register) .

TDT
See Trace Debug Tools

Test Access Port
The port used to access a device's TAP Controller. Comprises TCK, TMS, TDI, TDO and nTRST (optional).

TPA
See Trace Port Analyzer

Trace Port Analyzer
A particular variety of trace capture hardware.

See also Trace Capture Hardware.

Trace capture hardware
A device that is connected to the trace port on an ETM and converts the trace port signals into a form that the host computer and debugger can access. For example, it may convert it into packets on an ethernet network.

Trace Debug Tools A software product add-on to AXD that extends the debugging capability with the addition of real-time program and data tracing. TDT provides a user-friendly interface from which you can obtain an historical, non-intrusive trace of instruction flow and data accesses to help you identify a bug more efficiently than by using traditional debugging methods.

Index

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

L

M

R

S

T

W

