ARM® RMHost

User Guide

ARM

ARM DUI 0137A

ARM RMHost
User Guide

Release Information

The following changes have been made to this document.

Change history

Date	Issue	Change
December 2000	A	First release

Proprietary Notice

Contents
User Guide

Preface

This preface introduces the User Guide for ARM® RMHost. It contains the following sections:

About this book

This book describes how to configure the RealMonitor host controller, `RealMonitor.dll`, and how to debug RealMonitor-enabled applications using RMHost. This book documents only the host-side functionality of RealMonitor. For complete details on the target-side functionality, and how to prepare an RealMonitor-enabled application, see the *ARM RMTarget Integration Guide*.

Intended audience

This book is written for programmers who want to debug a time-critical application or RTOS using RMHost. It assumes that you are familiar with the process of debugging, and that you understand how RMTarget has been integrated into your application.

Using this book

This book is organized into the following chapters:

Chapter 1 ***Overview of RMHost***

Read this chapter for an introduction to RMHost, and for a description of the host-side functionality of RealMonitor, which is distinct from RMTarget. This chapter also describes the system requirements for using RMHost.

Chapter 2 ***Connecting to RMHost***

Read this chapter to see the procedure for connecting to the debug target using the *ARM eXtended Debugger* (AXD), and for details on selecting and configuring the RMHost controller. It also describes the connection messages you receive in AXD when you connect to RMHost.

Chapter 3 ***Debugging with RMHost***

Read this chapter for a description of the debugging features you can use, and the debugging restrictions you must be aware of, when connected to RMHost. It also describes all RMHost-related error messages you can see from within the debugger. It also describes the use of RMHost with *Trace Debug Tools* (TDT).

Typographical conventions

The following typographical conventions are used in this book:

bold Highlights interface elements, such as menu names. Also used for emphasis in descriptive lists, where appropriate.

italic	Highlights special terminology, denotes internal cross-references, and citations.
`typewriter`	Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
`typewriter`	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 and 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 from both ARM Limited and third parties that provide additional information on developing code for RMHost.

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

See also the ARM Frequently Asked Questions list at the ARM website.

ARM® publications

This book contains information that is specific to RMHost. Refer to the following books for related information:

- *Trace Debug Tools User Guide* (ARM DUI 0118)
- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)
- *ARM Firmware Suite User Guide* (ARM DUI 0136)
- *Multi-ICE User Guide* (ARM DUI 0048)
- *ARM RMTarget Integration Guide* (ARM DUI 0142)
- *CodeWarrior IDE Guide* (ARM DUI 0065)
- *AXD and armsd Debuggers Guide* (ARM DUI 0066)
- *ARM Architecture Reference Manual* (ARM DDI 0100) .

Other publications

Please refer to the following publications for additional information:

- E5903-97000, *Trace Port Analysis for ARM ETM User's Guide*, Agilent, 1999.
- E3459-97002, *Emulation for the ARM7/ARM9 User's Guide*, Agilent, 1999.

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

Feedback

ARM Limited welcomes feedback on both RMHost, and its documentation.

Feedback on RMHost

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

- details of the release you are using
- details of the host and target you are running on
- a small standalone sample of code that reproduces the problem
- a clear explanation of what you expected to happen, and what actually happened
- the commands you used, including any command-line options
- sample output illustrating the problem
- the version string of the JTAG unit, including the version number and date.

Feedback on this book

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

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

General suggestions for additions and improvements are also welcome.

Chapter 1
Overview of RMHost

This chapter introduces RMHost. It describes the host-side functionality of ARM RealMonitor, and how it is distinct from RMTarget. It also describes the system requirements for using RMHost.

This chapter contains the following sections:

1.1 About RMHost

RMHost is the host component of RealMonitor. It allows you to perform nonstop debugging on a RealMonitor-enabled application in a real-time environment. That is, when using RMHost, you can perform certain debugging operations on a foreground application while the processor continues to run (and service interrupts) in the background.

The functionality of RMHost is contained in the RMHost controller, `RealMonitor.dll`, which provides a standard *Remote Debug Interface* (RDI) to the debugger. The debugger communicates with the debug target using the RMHost controller, and communication between RMHost and RMTarget is governed by the RealMonitor protocol. These component parts of RealMonitor, and the connection between them, are shown in Figure 1-1.

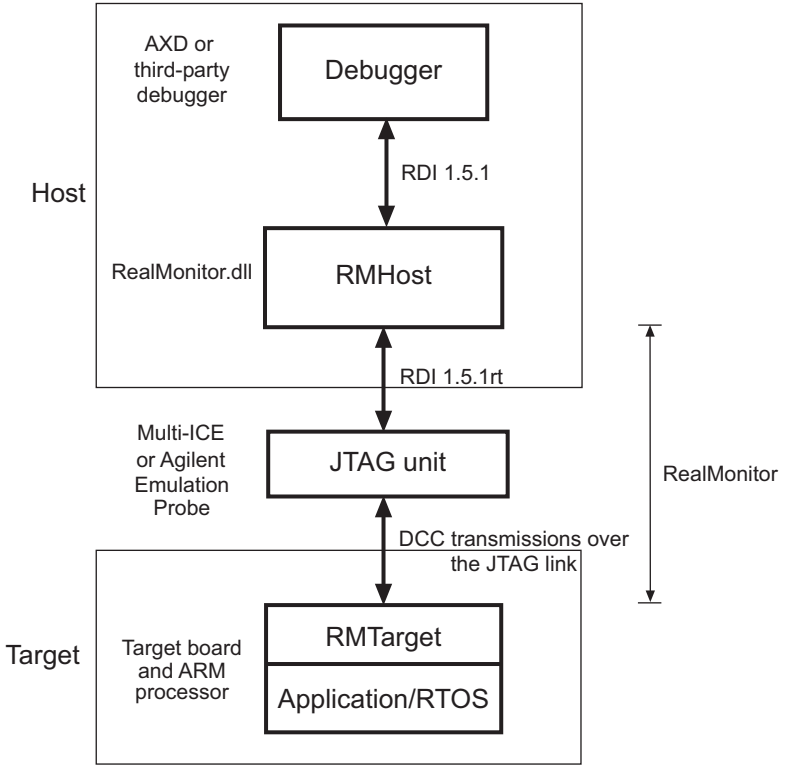

Figure 1-1 RealMonitor components

The RMHost controller converts RDI function calls into RealMonitor protocol packets, and uses the chosen JTAG unit to send these over the *Debug Communications Channel* (DCC) of the ARM processor. Additionally, RMHost uses the signals generated by the JTAG unit to control operation of RMTarget. For these reasons, the software that drives the JTAG unit must comply with RDI 1.5.1rt.

You must configure the RMHost controller when you first request connection to the target, which must already be running. You must then choose the supported JTAG unit to be used to communicate with, and control, the target. For complete details on the procedure for selecting and configuring the RMHost controller, see *Procedure for connecting to RMHost using AXD* on page 22.

1.2 RMHost requirements

This section describes the requirements necessary to use RMHost. RMHost works with any application that has been integrated with RMTarget. For details on building RMTarget, and integrating it with an application or *Real-Time Operating System* (RTOS), see the *ARM RMTarget Integration Guide*.

To run RMHost, your system must contain the following:

RMHost package

This is contained on the ARM RealMonitor CD-ROM. It must be installed on your system as described in the installation guide that accompanies the RealMonitor 1.0 CD-ROM.

Debugger You can use the following with RMHost:

- *ARM eXtended Debugger* (AXD), as provided with ADS version 1.1 or later.

  Note

  RMHost is not supported when using ADW.

- Any other debugger that conforms to RDI 1.5.1rt, and supports the *Real-Time* (RT) features used by RMHost. See the documentation that accompanies your debugger for details on the features it supports.

Platform The RMHost controller supports the following platforms:

- Windows 95
- Windows 98
- Windows 2000
- Windows NT 4.0
- HP-UX 10.20
- Solaris 2.6
- Solaris 7.0.

For details on the target hardware supported by RealMonitor, see the section on RealMonitor system requirements in the *ARM RMTarget Integration Guide*.

Chapter 2
Connecting to RMHost

This chapter describes the procedure for connecting to the debug target using AXD, including details on selecting and configuring the RMHost controller. It also describes the connection messages you receive in AXD when you connect to RMHost.

This chapter contains the following sections:

2.1 Procedure for connecting to RMHost using AXD

This section describes the steps you must follow to connect to RMHost using AXD 1.1. If you are using another debugger, see its accompanying documentation for details on how to perform these steps using the method that is specific to your debugger.

To debug a RealMonitor-enabled image, you must connect with RMHost while your program is executing. It is therefore important that you follow the steps of this procedure in the order presented.

Caution

If you are using the Multi-ICE® sever with RMHost, you must ensure that it is not autoconfigured because this causes the target board to be reset, and disrupts any running program.

When the Multi-ICE server is not preloaded and preconfigured, you might be given the option to restart the Multi-ICE server when you attempt to connect to the Multi-ICE DLL. In this case, you must click **No** because this would cause an autoconfiguration. Similarly, you must not configure the Multi-ICE server using the **Auto-configure** option.

To configure the Multi-ICE server for use with RMHost, you must create a configuration file that can be loaded into the Multi-ICE server, as follows:

1. Start the Multi-ICE server, and select **Auto-Configure** from the **File** menu. The file `Autoconf.cfg` is created (or updated) in the root directory of your Multi-ICE installation.
2. Make a copy of the file `Autoconf.cfg`, and give the copy an arbitrary name. Renaming this file ensures that it is not overwritten in a future Multi-ICE session.

After you have created and renamed your configuration file, you must load it into your Multi-ICE server using the **Load Configuration...** option from the **File** menu. You must do this every time you load the Multi-ICE server for use with RMHost.

Note

Steps 1 and 2 of the following procedure describe how to connect to RMHost if your image is not preloaded into ROM or Flash memory. If your image is preloaded, you must start at step 3.

For details on loading an image into Flash memory, see the chapter on using the *ARM Flash Utility* (AFU) in the *ARM Firmware Suite Reference Guide*.

To use RMHost with a RealMonitor-enabled image:

1. Start AXD, and configure the target as follows:
   a. Select **Configure Target** from the **Options** menu. The Choose Target dialog box is displayed.
   b. Select a supported JTAG unit DLL, such as `Multi-ICE.dll` version 2.0, in the Choose Target window, as shown in Figure 2-1. If this is not present in the list, click **Add**, and find the required DLL. (If DLL files do not appear, use Windows Explorer to ensure that files of extension `.dll` are not hidden from view.) The Multi-ICE DLL is found in the root directory of the Multi-ICE installation. If you have Multi-ICE 2.0 hardware, and you have not used Multi-ICE as a target in the past, click **Configure**. Otherwise, click **OK**. If you are using another JTAG unit, see its accompanying documentation for details on configuration.

Note

For details on configuring the Multi-ICE target, see the *Using Multi-ICE with Debuggers* chapter of the *Multi-ICE User Guide*.

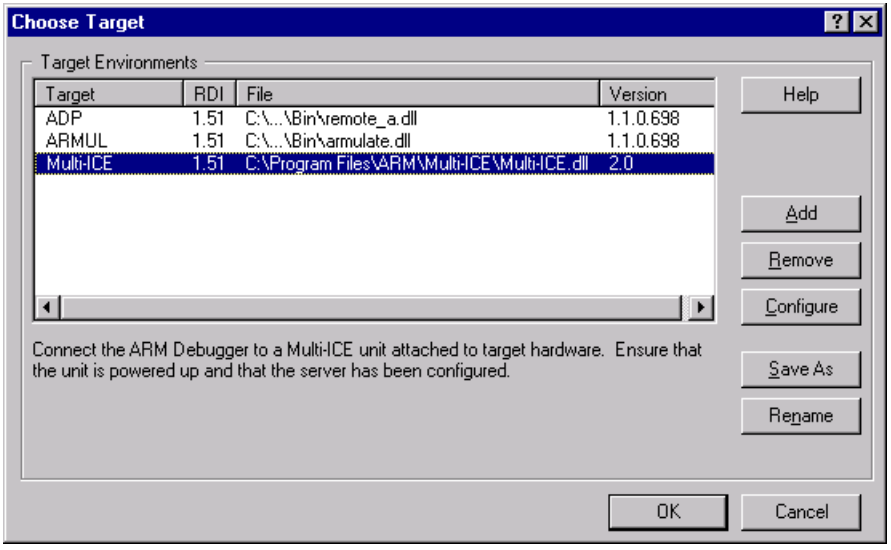

Figure 2-1 Selecting the Multi-ICE target

2. Load the RealMonitor-enabled image into your target board by selecting **Load Image** from the **File** menu, and execute it by selecting **Go** from the **Execute** menu.

Note

If **Load Image** is disabled, you must stop the currently running application by selecting **Stop** from the **Execute** menu.

The image might stop at `main()` if the program contains this function. In this case, select **Go** from the **Execute** menu to continue from the breakpoint.

3. While your program is running, configure the interface as follows:
   a. Select **Configure Interface** from the **Options** menu.
   b. Select the General tab.
   c. In the **Target connection** drop-down menu, select the option `ATTACH: Connect according to target properties`, as shown in Figure 2-2. Click **OK**.

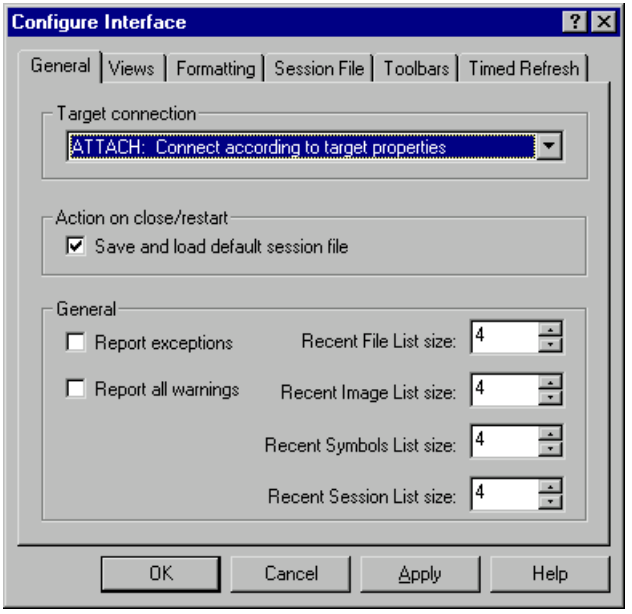

Figure 2-2 Configure interface dialog box

4. Select the RealMonitor target as follows:
   a. Select **Configure Target** from the **Options** menu. The Choose Target dialog box is displayed.
   b. Select `RealMonitor.dll` as the target DLL, as shown in Figure 2-3 on page 25.

If not present in the list, click **Add** and select `RealMonitor.dll` from the `Bin` directory of the ADS installation. (If DLL files do not appear, use Windows Explorer to ensure that files of extension `.dll` are not hidden from view.) The Choose Target dialog box is redisplayed.

When you select `RealMonitor.dll`, AXD displays the following message:

```
Connect the ARM Debugger to a target board using a unit such as Multi-ICE.
Boards require RMTarget software.
```

Step 5 describes how to do this.

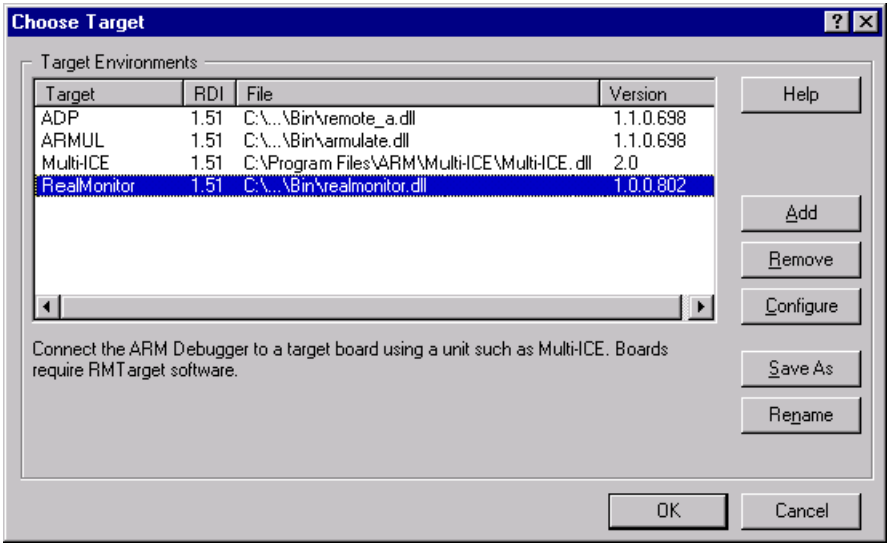

Figure 2-3 Choose Target dialog box

5. Click **Configure**. The RealMonitor Configuration dialog box is displayed, an example of which is shown in Figure 2-4 on page 26.

Note

If you have previously configured RMHost using the RealMonitor Configuration dialog box, you need to repeat the configuration process only if the configuration details have changed. If no details have changed, you can proceed to step 10.

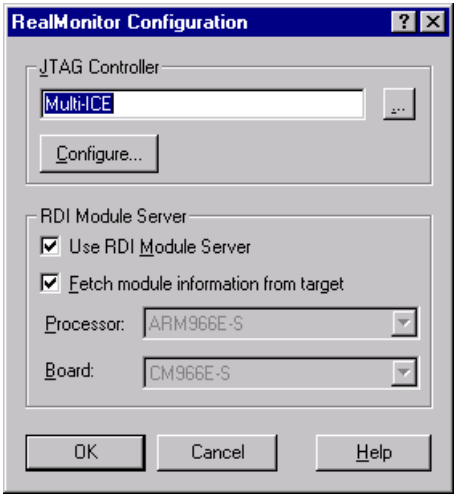

Figure 2-4 RealMonitor Configuration dialog box

Select an RDI 1.5.1rt-compliant JTAG controller DLL (see *RMHost requirements* on page 14), using one of the following methods:

- If present in either the `Bin` directory of the ADS installation, or in one of the `PATH` environment variable directories, type in the name of the DLL in the JTAG Controller field (with or without the `.dll` extension), such as `Multi-ICE`, as shown in Figure 2-4.
- If not present in these directories, browse by clicking **...**. A search dialog box is displayed, as shown in Figure 2-5 on page 27, where you must find and select the desired DLL. (If DLL files do not appear, use Windows Explorer to ensure that files of extension `.dll` are not hidden from view.) The Multi-ICE DLL is found in the root directory of the Multi-ICE installation.

  Note

  If the DLL you select is not supported by RealMonitor, an error message is displayed indicating that the chosen JTAG unit does not support RT extensions (see *Error messages* on page 37).

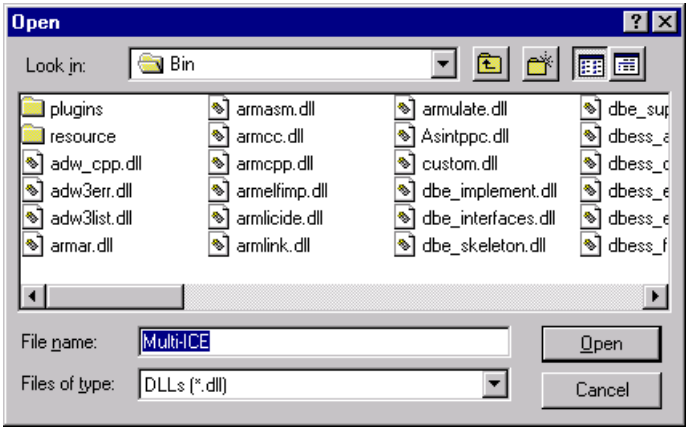

Figure 2-5 JTAG controller search dialog box

6. Click **Configure...** to configure the JTAG controller you have selected. For details on configuring Multi-ICE 2.0, see the *Using Multi-ICE with Debuggers* chapter of the *Multi-ICE User Guide*. If you are using another JTAG controller, see its accompanying documentation for details on configuration.

   Note

   Optionally, you can enable tracing if you have ARM *Trace Debug Tools* (TDT) installed. For details on enabling tracing, see the appendix on setting up the trace software in the *Trace Debug Tools User Guide*.

   When you have successfully configured the JTAG controller, the RealMonitor Configuration dialog box is redisplayed.

7. Optionally, you can enable use of the RDI module server. The RDI module server implements the host-side of the *Self-Describing Module* (SDM) extension to the RDI (see the build option `RM_OPT_SDM_INFO` in the building chapter of the *ARM RMTarget Integration Guide*).

   If you do not want to enable this feature, deselect **Use RDI Module Server**, and proceed to step 9.

   Enabling the RDI module server allows you, when connected to the RMHost controller, to view the entries in the registers view (of AXD) for any additional devices you have installed in your target system, such as coprocessors and interrupt controllers. You can view these entries by selecting **Registers** from the **Processor Views** menu.

   To enable use of the module server, select **Use RDI Module Server**. The option **Fetch module information from target** is no longer grayed out.

Caution

The module server does not work with the RMHost controller if you disabled support for the ExecuteCode packet when you built RMTarget. See the description of the `RM_OPT_EXECUTECODE` build option in the building the RMTarget chapter of the *ARM RMTarget Integration Guide*.

8. Provide the module server with information about the target system.

   If RMTarget was built with this information embedded, select **Fetch module information from target** and click **OK** to obtain this information (see the build option `RM_OPT_SDM_INFO` in the building chapter of the *ARM RMTarget Integration Guide*). If you select this option, the fields **Processor** and **Board** are grayed out, as shown in Figure 2-4 on page 26.

   If no information is returned when you select the option **Fetch module information from target** and click **OK**, you must provide information as follows:

   a. Deselect the the option **Fetch module information from target**.

   b. From the drop-down list in the Processor field, select the processor core being debugged, such as `ARM966E-S`. The list consists of processors supported by the RDI module server, including a blank entry that you must select if the system being debugged is not included in the list. See *RMHost requirements* on page 14 for details on hardware supported by the RealMonitor host controller.

   c. From the drop-down list in the Board field, select the board you are using, such as `CM966E-S + Integrator/AP`. The list consists of boards supported by the RDI module server, including a blank entry that you must select if the system being debugged is not included in the list. See *RMHost requirements* on page 14 for details on hardware supported by the RealMonitor host controller.

      Note

      It is possible to specify only a processor without specifying a board. In this case, only the processor state (coprocessor 15) is displayed by AXD when the module server is enabled. If you specify only a board, and no processor, only the state of the board peripherals is displayed by AXD when the module server is enabled.

9. Click **OK**.

Note

If RMHost detects an error in the configuration you have selected, an error message is displayed. If this happens, you can click one of the following:

OK Stores the configuration, although the error can prevent you from using RMHost.

Cancel Does not store the configuration, and you are returned to the RealMonitor Configuration dialog box.

See *Error messages* on page 37 for a description of RMHost-related errors.

If configuration is successful, the selected configuration is stored, and the Choose Target dialog box is displayed.

10. Click **OK**. You are ready to begin debugging your application (see Chapter 3 *Debugging with RMHost*).

Note

A dialog box indicating that the processor is still executing might be displayed. If this happens, click **No** to indicate that you do not want execution to stop.

Caution

When switching targets using AXD, a dialog box is displayed that requests whether you want to reload the last image. In this case, you must click **No** because reloading the last image corrupts the currently running image.

If you have disabled write access to the *program counter* (pc) in the register descriptor block, the message shown in Figure 2-6 is displayed (see *CPU register access* on page 34).

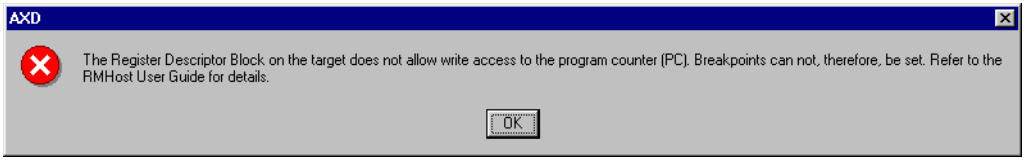

Figure 2-6 No pc write access message

If you have disabled read access to the pc in the register descriptor block, the message shown in Figure 2-7 on page 210 is displayed (see *CPU register access* on page 34).

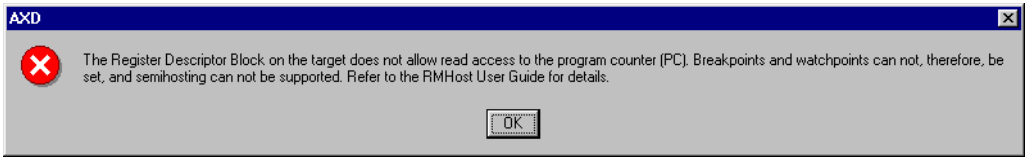

Figure 2-7 No pc read access message

2.2 AXD connection messages

When you connect the debugger to RMHost, messages are displayed in the RDI Log window of AXD, as shown in Example 2-1.

If you are using a debugger other than AXD, see the documentation that accompanies your debugger for details on where information is displayed when you connect to a new target.

Example 2-1 Example of RMHost connection messages

```
ARM RMHost 1.0 (ARM Ltd. RM1.0) [XXX]
ARM Multi-ICE Release 2.0. Copyright (c) ARM Limited 1998-2000.
Connected to TAP 0, ARM966E-S on Server "localhost", Little-Endian target.

ARM RMTarget A2.0 (ARM Ltd)
ARM RDI 1.5.1 -> ASYNC RDI Protocol Converter v1.1.Copyright (c) ARM Limited 2000.
```

The five lines of information shown in Example 2-1 indicate the following:

Line 1 The product name and version of RMHost, where *XXX* is the three-digit build number. You must quote this number when reporting any problems with RMHost.

Lines 2 and 3

Information that is specific to the JTAG unit, in this case Multi-ICE.

Line 4 The product name of RMTarget and version number of the RealMonitor protocol. The supplier for the target is also shown, which is decoded from the character at the start of the version number. Currently, the only supported supplier is ARM Ltd., as indicated by `A`.

Line 5 Generated by AXD. If you have enabled the RDI module server, some additional module-specific information might be displayed.

Chapter 3
Debugging with RMHost

This chapter describes all the debugging features you can use in the real-time debugging environment available when you are connected to RMHost. It also describes the restrictions to debugging a RealMonitor-enabled application, and includes descriptions of RMHost-related error messages.

It also describes the use of RMHost with *Trace Debug Tools* (TDT). It contains the following sections:

3.1 Debugging with RMHost

When connected to RMHost, the AXD user interface is identical to that when connected to other debug targets such as Multi-ICE. (For details on connecting to RMHost, see Chapter 2 *Connecting to RMHost*.) You can display, for example, memory windows, register contents, and variables.

This section describes the standard debugging features you can use when connected to RMHost, and how to use these features when connected to RMHost as the debug target:

- *Nonstop debug*
- *Background memory access* on page 33
- *CPU register access* on page 34
- *Nonstop startup* on page 34
- *Background setting and clearing of breakpoints and watchpoints* on page 35
- *Profiling* on page 35
- *Data messaging* on page 35
- *Semihosting* on page 35.

Note

The debugging features supported by RMHost are supported by any debugger that is RDI 1.5.1-compliant (see *RMHost requirements* on page 14).

3.1.1 Nonstop debug

When you are debugging with Multi-ICE, for example, and you stop the target using breakpoints, the processor is halted, and no application can run in the foreground or background.

Nonstop debug, which is available when connected to RMHost, refers to when the foreground application stops when a breakpoint is hit, and the processor continues to run. This allows system-critical tasks, such as IRQ and FIQ handlers, to continue to run as you debug your foreground application.

3.1.2 Background memory access

In addition to being able to access memory when your application is stopped (as with Multi-ICE, for example), memory can also be read or written to while your application is running.

Therefore, the memory and disassembly windows in the debugger you are using can be opened and refreshed while your application is running. You can also view the contents of non-automatic variables, that is, variables such as global variables and statics that are not allocated to registers or the stack. However, this ability can be restricted while debugging.

Memory can also be written to at any time. However you must be aware that read/write operations might not be atomic to the application. Therefore, if a large structure is viewed, for example, it might not be a consistent snapshot because data at the end of the structure is likely to have been read later than data at the start of the structure. The same problem exists with writes, so you must take care when updating such a structure when consistency between the elements is required.

If RMTarget has not been built with support for word accesses, for example, RMHost uses the supported access size (see `RM_OPT_WRITEWORDS` and `RM_OPT_READWORDS` in the building chapter of the *ARM RMTarget Integration Guide*). For byte and halfword accesses, RMTarget executes multiple memory-access instructions per word of data.

Note

This use of multiple memory-access instructions might cause unexpected results when accessing certain peripherals, so be sure to enable those build options in RMTarget that represent the access types supported by your peripherals.

While executing these instructions, your application is not running, but interrupts might still be serviced. It is therefore essential that you take care when writing to variables used by interrupt service routines.

RealMonitor also uses writes to memory to set software breakpoints. It is therefore possible that when setting a breakpoint on an interrupt service routine, the routine might execute before the breakpoint is fully set, causing the routine to malfunction (without causing an undefined instruction or breakpoint). If this happens, an error message is not necessarily displayed.

To work around this situation, RMHost always uses the ExecuteCode command to ensure that writes of a single word (or halfword, if using Thumb), such as a breakpoint instruction, are always atomic. That is, they use a single STR instruction (or STRH, if using Thumb). This is not done for larger structures.

Note

This workaround is performed only if you have built RMTarget to support the ExecuteCode packet (see RM_OPT_EXECUTECODE in the building chapter of the *ARM RMTarget Integration Guide*).

It is recommended that you build RMTarget with support for either or both of the following:

- word and/or halfword writes (see RM_OPT_WRITEWORDS and RM_OPT_WRITEHALFWORD in the building chapter of the *ARM RMTarget Integration Guide*)
- the ExecuteCode packet (see RM_OPT_EXECUTECODE in the building chapter of the *ARM RMTarget Integration Guide*).

3.1.3 CPU register access

CPU registers can be read while your application is stopped. It is not possible to read/write CPU registers while your application is running.

Access to specific registers can be restricted using the register accessibility block built into RMTarget (see the RM_RegisterAccess section of the API chapter in the *ARM RMTarget Integration Guide*). If this is done, certain registers can become unreadable or unwritable. If marked as unreadable, AXD does not return an error, but it displays register values as zero.

Read and write access must not be removed from the *program counter* (pc) if start-stop debugging (such as the ability to set breakpoints and stop the application) is to be supported. Similarly, access to the unbanked registers must not be removed if semihosting is to be supported.

3.1.4 Nonstop startup

When your application is running, and you establish connection to RMHost, your target application is typically stopped. With RMHost, you can select a mode that allows your application to continue to run when you establish connection to RMHost. If you connect using this method, you can then check the state of your application without having to stop it. You can do this using background memory accesses, which are supported by RMHost (see *Background memory access* on page 33).

This feature is enabled when you select the option ATTACH: Connect according to target properties. For more details, see the step on configuring the interface in *Procedure for connecting to RMHost using AXD* on page 22.

3.1.5 Background setting and clearing of breakpoints and watchpoints

When connected to RMHost, you can set and clear breakpoints and watchpoints without stopping your application. You can also set and clear breakpoints and watchpoints when your application is in a stopped state.

3.1.6 Profiling

RMHost supports a low-intrusion code profiling mechanism. This mechanism works only if you enabled profiling when you built RMTarget (see the build option `RM_OPT_GETPC` in the building chapter of the *ARM RMTarget Integration Guide*). For details on how sample-based profiling is performed by the GetPC command, see the description of GetPC in the RealMonitor protocol chapter of the *ARM RMTarget Integration Guide*.

For details on code profiling using AXD, see the AXD facilities chapter of the *AXD and armsd Debuggers Guide*.

Note

The profiling interval specified by the debugger is ignored by RMHost. In AXD, the interval is specified in the Load Image dialog box.

3.1.7 Data messaging

Your application can use RealMonitor to buffer and send messages to the debugger. RMHost passes these messages to the Debug Comms Channel processor view in AXD. This feature works only if you have enabled data messaging when you built RMTarget (see the build option `RM_OPT_DATALOGGING` in the building chapter of the *ARM RMTarget Integration Guide*).

For details on the Debug Comms Channel processor view, see the AXD facilities chapter of the *AXD and armsd Debuggers Guide*.

3.1.8 Semihosting

RMHost supports the use of the ARM semihosting interface. This support works only if you have enabled the SWI handler when you built RMTarget (see the build option `RM_OPT_SEMIHOSTING` in the building chapter of the *ARM RMTarget Integration Guide*).

If you are using semihosting, and you stop your application, you might receive the following message:

```
Semihosting write call interrupted (data will be lost)
```

This message is displayed when the application is writing to the console, and you stop your application at the same time RMHost is performing a write. In this case, the data that has not been written is lost, and is not displayed.

For complete details on semihosting with an ARM target, see the semihosting chapter of the *ADS Debug Target Guide*.

3.2 Error messages

This section describes all error messages you might receive when connecting to, or using, RMHost.

Note

Error messages can be generated by both the debugger and the JTAG unit you are using. Errors that come from the JTAG unit (such as Multi-ICE) are prefixed in the debugger by the string `(JTAG)`. For a description of these errors, see the documentation that accompanies the JTAG unit you are using.

Errors from the RDI Module Server are prefixed with the string `(MSVR)` if you have enabled the module server when configuring RMHost (see *Procedure for connecting to RMHost using AXD* on page 22).

The error messages are as follows:

`Cannot use RMHost as JTAG controller`

This is displayed if you attempt to select `RealMonitor.dll` as the JTAG controller.

`Configured JTAG controller DLL not compatible with RMHost`

The configured JTAG controller DLL loaded successfully, and implements RDI, but it does not support a sufficient level of RDI to be used by RMHost. In this case, you must upgrade your JTAG controller software to be RDI 1.5.1rt-compliant.

`Configured JTAG controller does not support RT extensions`

JTAG controllers used by RMHost must support the *real-time* (rt) extensions to RDI. In this case, you must upgrade your JTAG controller software to be RDI 1.5.1rt-compliant.

`Configured JTAG controller is not a valid RDI target`

The DLL you have chosen for the JTAG controller does not implement RDI.

`Data link timeout error`

A link timeout typically occurs when the target is either not responding, or is responding very slowly. If the target is not responding at all, there is an error in the target. In this case, it is recommended that you either stop the target and examine its state directly using JTAG debugging, or reset the target.

This error might also be displayed when any of the following occurs:

- You connect to a remote target, that is, one that is not connected to the local machine. In this case, a timeout typically results from a network error.
- The target is built for polled mode, and the application is not calling `RM_PollDCC()`. See the description of this function in the API chapter of the *ARM RMTarget Integration Guide*.
- The communications channel interrupts on the target have a very low priority.

`DBE Warning 00030 on object 'Oscillator': Could not write the item requested (possibly read-only)`

An error of this type indicates that a given register (`Oscillator`, in this case) could not be written to for a variety of reasons.

`DBE Warning 00031 on object 'Id': Could not read from the processor register requested`

An error of this type indicates that a given register (`Id`, in this case) could not be read for a variety of reasons.

Note

The register value is displayed as zero when it cannot be read. This error is displayed only in the Debug Log Window.

`Failed to read or write target memory`

The RMHost controller was attempting to perform an operation that involved reading or writing memory, and this failed. (For example, there might have been a Data Abort while attempting to set a software breakpoint.)

`JTAG controller is not reporting a comms channel`

The processor being debugged does not support a debug communications channel, and therefore, RMHost cannot connect to it.

`JTAG controller is not reporting an 'ARM' module`

The target does not include an ARM processor.

`JTAG controller is not reporting non-stop debug capability`

The JTAG controller would have to put the processor into debug state to allow connection to RMHost, and therefore, connection must be aborted. Check that your JTAG controller software is RDI 1.5.1rt-compliant.

`Memory map forbids reading from location`

Either you or the debugger attempted to read a location, but the memory map on the target forbids the reading of locations in that region.

`Memory map forbids setting of breakpoint/watchpoint`

Either you or the debugger attempted to set a breakpoint or watchpoint, but the memory map on the target forbids the setting of watchpoints or breakpoints on that region. (The debugger might attempt to set a breakpoint when single-stepping.) For more details, see `rm_MemoryMap` in the API chapter of the *ARM RMTarget Integration Guide*.

Note

For details on memory maps, see the *ARM RMTarget Integration Guide* for a description of `rm_Memory` in the API chapter, and the pointer to memory descriptor block in the RealMonitor protocol chapter.

`Memory map forbids writing to location`

Either you or the debugger attempted to write a location, but the memory map on the target forbids the writing of locations in that region.

`Message received from target was not handled by RMHost`

The RealMonitor protocol includes messages sent from the target to the controller, such as `RM_Msg_HardBreak` (RealMonitor has stopped at a breakpoint). This error message indicates that the target sent a message, but the controller did not understand it, and usually indicates an error in RMTarget (see the section on target-to-host controller messages in the RealMonitor protocol chapter of the *ARM RMTarget Integration Guide*).

`RealMonitor not compatible with RDI target`

The RDI target (usually Multi-ICE) does not support the features necessary to run RMHost. Either an invalid target has been chosen, or the RDI target must be upgraded.

`Remote target is not executing`

The target processor is in debug state, and therefore, it cannot be debugged by RMHost.

`RMHost buffer overflow`

An internal buffer inside the RMHost controller has overflowed. As with an internal error, this error must be reported to your supplier.

`RMHost could not load the specified JTAG controller`

This is displayed when the JTAG controller DLL could not be loaded.

`RMHost internal error`

Indicates some consistency failure inside the RMHost controller. You must report this error to your supplier, along with a description of the events leading up to the error.

`RMHost protocol error (internal error)`

The RMHost controller detected that it was about to violate the RealMonitor protocol, and aborted the operation. This must be treated in the same way as an internal error, and must be reported to your supplier.

`RMTarget does not support an expected feature`

The RMHost controller sent a command to the monitor that was not understood. This typically indicates an error in the RMTarget build, because the controller checks that a command is supported before using it.

`RMTarget entered panic state (debug event while stopped)`

The monitor detected a debug event (such as a breakpoint) while it was already stopped, and has gone into panic state. This should not occur during normal operation, and usually indicates a user error. For example, this might occur if you have set a breakpoint on some code that is used by an interrupt routine. The target must be reset.

`RMTarget protocol error`

The controller has detected a violation of the RealMonitor protocol by the target. This usually indicates a target fault, which can typically be resolved by resetting it.

`RMTarget version does not match RMHost version`

As part of initialization, the controller checks that the version number of the target matches its own version number. If they do not match, the controller cannot connect to the target. The version number is used in this way to protect against changes in the RealMonitor protocol between versions.

`Unknown RMTarget error`

RMTarget is reporting an error that is not understood by RMHost. See the description of the pointer to the error block in the *ARM RMTarget Integration Guide*.

3.2.1 Other messages from RMHost

In addition to the messages described in *Error messages* on page 37, you can also receive a message similar to that shown in Figure 3-1 when either of the following occurs:

- you write to the CPSR
- the debugger writes to the CPSR after an image load.

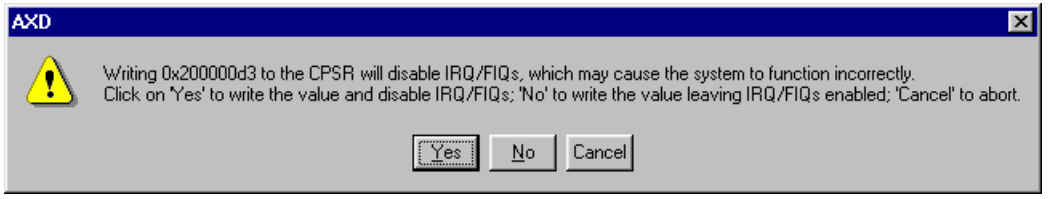

Figure 3-1 IRQ message

In this case, RealMonitor has detected that either you or the debugger are attempting to disable either IRQs or FIQs. Because RMTarget is IRQ-driven, and because one of the purposes of RealMonitor is to allow debugging without disabling FIQs, RMHost produces this warning before allowing the write to occur.

Caution

Failure to choose the correct response to this prompt can cause the target to crash, or the debug session to lock up.

If you receive this message, you must click one of the following:

Yes Write the value and disable IRQs. The specified interrupts are disabled. If there is no code in your application to re-enable IRQs, RMTarget cannot respond to some debug requests.

You must click **Yes** if your application code relies on IRQs being disabled during initialization, such as when it needs to install its own IRQ handler. In this case, the application enables IRQs after initialization.

No Write the value specified, with the exception of the IRQ/FIQ disable bits. These interrupts remain enabled.

Note

The values shown in AXD will be incorrect in this case.

You must click **No** if your target application does not enable IRQs itself. This allows RMTarget to service interrupts.

Cancel Abort. RMHost returns an error to the debugger.

Note

In general, if your application enables interrupts itself, you must start it with interrupts disabled. Otherwise, it is typically safe to start it with interrupts enabled.

3.3 Using RMHost with Trace Debug Tools

RMHost can work with the ARM *Trace Debug Tools* (TDT) version 1.1 or later. The system configuration is illustrated in Figure 3-2.

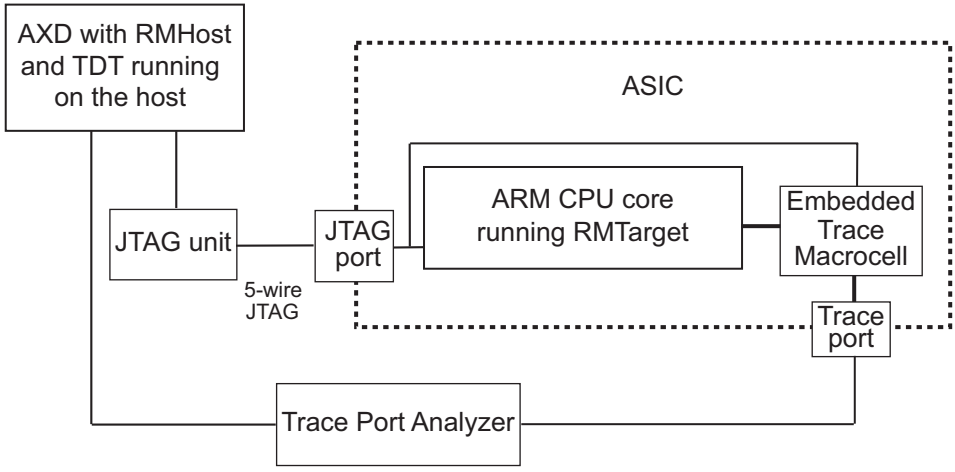

Figure 3-2 Using RMHost with TDT

Note

The ARM CPU core shown in Figure 3-2 can be any of the RealMonitor-supported target processors (see the section on system requirements in the *Introduction* chapter of the *ARM RMTarget Integration Guide*). For details on porting RMTarget to a new board or processor, see the section on porting RMTarget in the *Introduction to RMTarget* chapter of the *ARM RMTarget Integration Guide*.

When you are using AXD with both TDT and RMHost, you can perform tracing on your application, and then actively debug your foreground application using RMHost, based on the results returned to the Trace window. If you are using TDT without RMHost, you can debug your application based on the trace results, but this causes the processor to stop. Therefore, using the two products together provides you with a more robust debugging environment than using each product separately.

The system requirements are the same, except additional hardware is required to use TDT. These requirements are described in the section on setting up the TDT hardware in the *Getting Started* chapter of the *Trace Debug Tools User Guide*.

This section describes the following:

- *Configuring AXD to enable tracing with RMHost* on page 314
- *Performance of RMHost with TDT* on page 314.

3.3.1 Configuring AXD to enable tracing with RMHost

The process of connecting to a target system using both RMHost and TDT is similar to connecting to a target system using only RMHost. The only difference is that you must enable tracing when you configure your JTAG controller to be used with `RealMonitor.dll`, as described in step 6 of *Procedure for connecting to RMHost using AXD* on page 22. For details on enabling tracing, see the appendix on setting up the trace software in the *Trace Debug Tools User Guide*.

3.3.2 Performance of RMHost with TDT

When using TDT with Multi-ICE only, and not RMHost, TDT cannot read the target memory without halting the processor. However, it must be able to view the instructions that were executed by the processor to decode the trace data. It therefore implements an image cache in the host, and fetches the instructions from this cache, rather than from the target.

Note

If your image is already loaded onto the target, you can select **Load Debug Symbols** from the **File** menu to place the image into the image cache (even while your application is currently running).

Caution

If you are using self-modifying code, you must disable the image cache in the debugger because when the image cache is in use, the debugger cannot detect any modifications the code makes to itself. To disable the image cache in AXD:

1. Select **Debugger Internals** from the **System Views** menu.
2. Change the value of the variable `image_cache_enable` from `0x01` to `0x00`.

With RMHost, it is possible to read memory without halting the target, and TDT can decode the trace data by reading memory on the target. Therefore, a trace stream can be displayed without having to use the image cache.

However, because the link between the host and the target is limited by the JTAG connection, and the performance of the debug communications channel when using RMHost, decoding trace in this way is slow compared to using the image cache.

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.

G

H

I

J

M

N

P

R

S

T

U

W

