ARM® DS-5™

Version 5.6

Getting Started with DS-5

ARM®

ARM DUI 0478F (ID071411)

ARM DS-5
Getting Started with DS-5

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
June 2010	A	Non-Confidential	First release for DS-5
September 2010	B	Non-Confidential	Update for DS-5 version 5.2
November 2010	C	Non-Confidential	Update for DS-5 version 5.3
January 2011	D	Non-Confidential	Update for DS-5 version 5.4
May 2011	E	Non-Confidential	Update for DS-5 version 5.5
July 2011	F	Non-Confidential	Update for DS-5 version 5.6

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 DS-5 Getting Started with DS-5

Chapter 1
Conventions and feedback

The following describes the typographical conventions and how to give feedback:

Typographical conventions

The following typographical conventions are used:

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

`mono`space Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.

`monospace italic`

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

`monospace bold`

Denotes language keywords when used outside example code.

italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold Highlights interface elements, such as menu names. Also used for emphasis in descriptive lists, where appropriate, and for ARM® processor signal names.

Feedback on this product

If you have any comments and suggestions about this product, contact your supplier and give:

- your name and company

- the serial number of the product
- details of the release you are using
- details of the platform you are using, such as the hardware platform, operating system type and version
- a small standalone sample of code that reproduces the problem
- a clear explanation of what you expected to happen, and what actually happened
- the commands you used, including any command-line options
- sample output illustrating the problem
- the version string of the tools, including the version number and build numbers.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM DUI 0478F
- if viewing online, the topic names to which your comments apply
- if viewing a PDF version of a document, the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

ARM periodically provides updates and corrections to its documentation on the ARM Information Center, together with knowledge articles and *Frequently Asked Questions* (FAQs).

Other information

- *ARM Information Center*, `http://infocenter.arm.com/help/index.jsp`
- *ARM Technical Support Knowledge Articles*, http://infocenter.arm.com/help/topic/com.arm.doc.faqs
- *Support and Maintenance*, http://www.arm.com/support/services/support-maintenance.php
- *ARM Glossary*, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Chapter 2
ARM DS-5 product overview

The following topics give an overview of ARM® *Development Studio* (DS-5™).

Concepts

2.1 About DS-5

DS-5 is a professional software development solution for Linux-based systems and bare-metal embedded systems, covering all stages in development from boot code and kernel porting to application and bare-metal debug. It also includes performance analysis.

DS-5 includes:

- DS-5 Debugger.
- Eclipse for DS-5. An *Integrated Development Environment* (IDE) that combines the Eclipse IDE from the Eclipse Foundation with compilation and debug tools.
- Real-Time System Models.
- ARM Streamline™ Performance Analyzer.
- Dedicated examples, applications, and supporting documentation to help you get started with using the DS-5 tools.
- Debug hardware configuration utitlities for bare-metal.
- ARM Compiler tools for development of bare-metal embedded systems.
- GNU compilation tools for development of boot code and ARM Linux applications.

2.1.1 See also

Concepts

- *About Eclipse for DS-5* on page 2-3
- *About DS-5 Debugger* on page 2-4
- *About Real-Time System Models* on page 2-5
- *About ARM Compiler* on page 2-6
- *About GNU Compilation Tools* on page 2-7
- *About ARM Streamline Performance Analyzer* on page 2-8
- *About Debug hardware configuration utilities* on page 2-9.

Reference

- *Licensing and product updates* on page 4-4
- *Documentation* on page 4-5
- *Examples* on page 4-6.

Other information

- *DS-5 Knowledge Articles*, `http://infocenter.arm.com/help/topic/com.arm.doc.faqs/kiXXwMK1Sxk7vf.html`.

2.2 About Eclipse for DS-5

Eclipse for DS-5 is an *Integrated Development Environment* (IDE) that combines the Eclipse IDE from the Eclipse Foundation with the compilation and debug technology of the ARM tools. It also combines the GNU toolchain for ARM Linux targets.

Eclipse for DS-5 provides:

Project manager

This enables you to perform various project tasks such as adding or removing files and dependencies to projects, importing, exporting, or creating projects, and managing build options.

Editors These enables you read, write, or modify C/C++ or ARM assembly language source files.

Perspectives and views

These provide customized views, menus, and toolbars to suit a particular type of environment. DS-5 uses the C/C++ and DS-5 Debug perspectives.

2.2.1 See also

Tasks

- *ARM® DS-5™ Using Eclipse*:
  - Chapter 3 *Getting started with Eclipse*.

2.3 About DS-5 Debugger

DS-5 Debugger is a graphical debugger supporting end-to-end software development on ARM processor-based targets and *Real-Time System Models* (RTSMs). It makes it easy to debug Linux and bare-metal applications with comprehensive and intuitive views, including sychronized source and disassembly, call stack, memory, registers, expressions, variables, threads, breakpoints, and trace.

Using the Debug Control view you can single step through either at source level or instruction level and see the other views update as the code is executed. Setting breakpoints or watchpoints can assist you by stopping the application and enabling you to explore the behavior of the application. You can also use the Trace view on some targets to trace function executions in your application with a timeline showing the sequence of events.

You can also debug using the **DS-5 Command Prompt** command-line console.

2.3.1 See also

Tasks

- *ARM® DS-5™ Using the Debugger*:
  - Chapter 2 *Getting started with the debugger*.

Concepts

- *About Real-Time System Models* on page 2-5.

2.4 About Real-Time System Models

Real-Time System Models (RTSM) enable development of software without the requirement for actual hardware. The functional behavior of the model is equivalent to real hardware from a programmers view.

Absolute timing accuracy is sacrificed to achieve fast simulated execution speed. This means that you can use a model for confirming software functionality, but you must not rely on the accuracy of cycle counts, low-level component interactions, or other hardware-specific behavior.

DS-5 includes a Cortex™-A8 RTSM that is preconfigured to boot ARM Linux.

2.4.1 See also

Reference

- *RealView® Development Suite Real-Time System Models User Guide*, http://infocenter.arm.com/help/topic/com.arm.doc.dui0424-

2.5 About ARM Compiler

DS-5 includes a distribution of the ARM® Compiler tools.

These tools can be used to build applications and libraries suitable for bare-metal embedded systems, including the examples that are available in the DS-5 examples directory.

The ARM Compiler tools are located in *tools_directory*. You can use them to build your applications from either the command-line or within Eclipse.

Table 2-1 ARM Compiler tools

Tool	Description
`armar`	Librarian. This enables sets of ELF format object files to be collected together and maintained in archives or libraries. You can pass such a library or archive to the linker in place of several ELF files. You can also use the archive for distribution to a third party for application development.
`armasm`	Assembler. This assembles ARM and Thumb® assembly language sources.
`armcc`	Compiler. This compiles your C and C++ code. It supports inline and embedded assemblers, and also includes the NEON™ vectorizing compiler.
`armlink`	Linker. This combines the contents of one or more object files with selected parts of one or more object libraries to produce an executable program.
`fromelf`	Image conversion utility. This can also generate textual information about the input image, such as disassembly and its code and data size

2.5.1 See also

Tasks

- *Creating a new C or C++ project in Eclipse* on page 3-3.

2.6 About GNU Compilation Tools

DS-5 includes a distribution of the GNU Compilation Tools.

These tools can be used to build applications and libraries suitable for ARM Linux targets, including the example ARM Linux distribution that is available in the DS-5 examples directory. They are not suitable for building:

- bare-metal ARM targets
- ARM targets running any operating system other than ARM Linux
- non-ARM targets.

The GNU Compilation Tools are located in *tools_directory*. You can use them to build your applications from either the command-line or within Eclipse.

Table 2-2 GNU Compilation Tools

Tool	Description
arm-none-linux-gnueabi-ar	GNU librarian
arm-none-linux-gnueabi-as	GNU assembler
arm-none-linux-gnueabi-gcc	GNU c compiler
arm-none-linux-gnueabi-g++	GNU C++ compiler
arm-none-linux-gnueabi-ld	GNU linker

Getting Started with the GNU Compilation Tools is located in *documents_directory*\gcc.

2.6.1 See also

Tasks

- *Creating a new C or C++ project in Eclipse* on page 3-3
- *Building the Gnometris project from Eclipse* on page 3-4
- *Building the Gnometris project from the command-line* on page 3-5.

2.7 About ARM Streamline Performance Analyzer

ARM Streamline is a graphical performance analysis tool. Combining a kernel driver, target daemon, and an Eclipse-based user interface, it transforms sampling data and system trace into reports that present the data in both visual and statistical forms. Streamline uses hardware performance counters with kernel metrics to provide an accurate representation of system resources.

2.7.1 See also

Tasks

- *Performance analysis of threads application running on ARM Linux* on page 3-22.

Reference

- *ARM® DS-5™ Using ARM Streamline*, `http://infocenter.arm.com/help/topic/com.arm.doc.dui0482-/index.html`.

2.8 About Debug hardware configuration utilities

The debug hardware configuration utilities enable you to connect to the debug hardware unit that provides the interface between your development platform and your PC. The following utilities are provided:

Debug Hardware Config IP

Used to configure the IP address on a debug hardware unit.

Debug Hardware Update

Used to update the firmware and devices on a debug hardware unit.

Debug Hardware Configuration

Used to configure a debug hardware unit.

2.8.1 See also

Reference

- *ARM® DSTREAM™ and RVT™ Using the Debug Hardware Configuration Utilities*, http://infocenter.arm.com/help/topic/com.arm.doc.dui0498-/index.html.

Chapter 3
ARM DS-5 tutorials

The following tutorials show you how to run and debug applications using ARM® DS-5™ tools.

Tasks

3.1 Importing the example projects into Eclipse

Many tasks described in the documentation use the example projects provided with DS-5.

To use the example projects in Eclipse, you must first import them:

1. Launch Eclipse:
   - On Windows, select **Start → All Programs → ARM DS-5 → Eclipse for DS-5**.
   - On Linux, enter `eclipse` in the Unix bash shell.
2. ARM recommends that you create a new workspace for the example projects so that they remain separate from your own projects. To do this you can either:
   - Create a new workspace directory during the startup of Eclipse.
   - If Eclipse is already open, select **File → Switch Workspace → Other** from the main menu.
3. Select **Cheat Sheet...** from the **Help** menu.
4. Expand the **ARM -Eclipse for DS-5** group.
5. Select **Automatically Import the DS-5 Example Projects into the Current Workspace** from the list of ARM cheat sheets.
6. Click **OK**.
7. Follow the steps in the cheat sheet to import all the DS-5 example projects into your workspace.

When the examples are imported, you can optionally follow the remaining cheat sheet instructions to switch on working sets if required.

3.1.1 See also

Tasks

- *Building the Gnometris project from Eclipse* on page 3-4
- *Building the Gnometris project from the command-line* on page 3-5
- *Loading the Gnometris application on a Real-Time System Model* on page 3-6
- *Loading the Gnometris application on to an ARM Linux target* on page 3-7
- *Debugging Gnometris* on page 3-16
- *ARM® DS-5™ Using Eclipse*:
  - *Launching Eclipse* on page 3-3
  - *Creating a working set* on page 3-16
  - *Changing the top level element when displaying working sets* on page 3-19
  - *Deselecting a working set* on page 3-20
  - *Using the import wizard* on page 3-35.

Concepts

- *About Eclipse for DS-5* on page 2-3
- *Examples* on page 4-6
- *ARM® DS-5™ Using Eclipse*:
  - *About working sets* on page 3-15.

Reference

- *Examples* on page 4-6.

3.2 Creating a new C or C++ project in Eclipse

To create a new C or C++ Project:

1. Select **File → New → Project...** from the main menu.
2. Expand the **C/C++** group.
3. Select either **C Project** or **C++ Project**.
4. Select the type of project that you wish to create.
5. Click on **Next**.
6. Enter a project name.
7. Leave the **Use default location** option selected so that the project is created in the default directory shown. Alternatively, deselect this option and browse to your preferred project directory.
8. Select the type of project that you want to create.
9. Click on **Finish** to create your new project.

   The project is visible in the Project Explorer view.

3.2.1 See also

Tasks

- *ARM® DS-5™ Using Eclipse*:
  - *Creating a new C or C++ project* on page 4-4.

3.3 Building the Gnometris project from Eclipse

Gnometris is an ARM® Linux application that you can run and debug on your target. The supplied project does not contain the image binaries for the Gnometris application. To create the image, you must build the project.

To build the project:

1. Download the optional package, `Linux_distribution_example.zip`, containing the example Linux distribution project and the compatible headers and libraries from the ARM website or from the DS-5 installation media.
2. Import both the `gnometris` and `distribution` example projects from the relevant ZIP archive files into Eclipse.
3. Select the `gnometris` project in the Project Explorer view.
4. Select **Build Project** from the **Project** menu.

   The Gnometris example contains a `Makefile` to build the project. The `Makefile` provides the usual make rules: `clean`, `all`, and `rebuild`.

When you build the Gnometris project, it produces the following applications:

- A stripped version of the application containing no debug information. This is for downloading to the target.
- A larger sized version of the application containing full debug information for use by the debugger when debugging at the source level.

3.3.1 See also

Tasks

- *Importing the example projects into Eclipse* on page 3-2
- *Building the Gnometris project from the command-line* on page 3-5
- *Loading the Gnometris application on a Real-Time System Model* on page 3-6
- *Loading the Gnometris application on to an ARM Linux target* on page 3-7
- *Debugging Gnometris* on page 3-16
- *ARM® DS-5™ Using Eclipse*:
  - Chapter 4 *Working with projects*.

Reference

- *Examples* on page 4-6.

3.4 Building the Gnometris project from the command-line

Gnometris is an ARM® Linux application that you can run and debug on your target. The supplied project does not contain the image binaries for the Gnometris application.

To build the project:

1. Download the optional package, `Linux_distribution_example.zip`, containing the example Linux distribution project and the compatible headers and libraries from the ARM website or from the DS-5 installation media.
2. Extract both the `gnometris` and `distribution` example projects from the relevant ZIP archive files into a working directory.
3. Open the **DS-5 Command Prompt** command-line console or a Unix bash shell.
4. Navigate to `...\ARMLinux\gnometris`.
5. At the prompt, enter `make`. The example contains a `Makefile` to build the project. The `Makefile` provides the usual make rules: `clean`, `all`, and `rebuild`.

When you build the Gnometris project, it produces the following applications:

- A stripped version of the application containing no debug information. This is for downloading to the target.
- A larger sized version of the application containing full debug information for use by the debugger when debugging at the source level.

3.4.1 See also

Tasks

- *Building the Gnometris project from Eclipse* on page 3-4
- *Loading the Gnometris application on a Real-Time System Model* on page 3-6
- *Loading the Gnometris application on to an ARM Linux target* on page 3-7
- *Debugging Gnometris* on page 3-16.

Reference

- *Examples* on page 4-6.

3.5 Loading the Gnometris application on a Real-Time System Model

You can load the Gnometris application on to a *Real-Time System Model* (RTSM) that is running ARM Linux. An RTSM enables you to run and debug applications on your host workstation without using any hardware targets.

A preconfigured RTSM connection is available that automatically boots Linux, launches `gdbserver`, and then launches the application.

To load Gnometris:

1. Launch Eclipse.
2. Click on the Project Explorer view.
3. Expand the `gnometris` project folder.
4. Right-click on the launch file, `gnometris-RTSM-example.launch`.
5. In the context menu, select **Debug As**.
6. Select the `gnometris-RTSM-example` entry in the submenu.
7. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.

3.5.1 See also

Tasks

- *Importing the example projects into Eclipse* on page 3-2
- *Building the Gnometris project from Eclipse* on page 3-4
- *Building the Gnometris project from the command-line* on page 3-5
- *Debugging Gnometris* on page 3-16
- *ARM® DS-5™ Using the Debugger*:
  - *Configuring a connection to an RTSM model on page 3-3*.

Reference

- *Documentation* on page 4-5
- *Examples* on page 4-6
- *ARM® DS-5™ Using the Debugger*:
  - *Debug Configurations - Connection tab on page 11-60*
  - *Debug Configurations - Files tab on page 11-64*
  - *Debug Configurations - Debugger tab on page 11-68*
  - *Debug Configurations - Environment tab on page 11-74*.

3.6 Loading the Gnometris application on to an ARM Linux target

You can load the Gnometris application on to a target that is running ARM® Linux.

DS-5 provides preconfigured target connection settings that connect the debugger to `gdbserver` running on supported ARM architecture-based platforms.

To load an application:

1. Obtain the IP address of the target. You can use the `ifconfig` application in a Linux console. The IP address is denoted by the **inet addr**.
2. Boot the appropriate Linux distribution on the target.
3. Launch Eclipse.
4. Transfer the application and related files to the ARM Linux target, run the application, and then connect the debugger. There are several ways to do this:
   - On the Beagle board you can use a *Secure SHell* (SSH) connection with the *Remote System Explorer* (RSE) provided with DS-5 to set up the target and run the application. When the application is running you can then connect the debugger to the running target.
   - For other targets you can use an external file transfer utility such as `PuTTY`.

3.6.1 See also

Tasks

- *Using an SSH connection to set up and run Gnometris on an ARM Linux target* on page 3-8
- *Connecting to the Gnometris application that is already running on a ARM Linux target* on page 3-13
- *Debugging Gnometris* on page 3-16.

Reference

- *Documentation* on page 4-5
- *Examples* on page 4-6
- *ARM® DS-5™ Using the Debugger*:
  - *Debug Configurations - Connection tab on page 11-60*
  - *Debug Configurations - Files tab on page 11-64*
  - *Debug Configurations - Debugger tab on page 11-68*
  - *Debug Configurations - Environment tab on page 11-74*
  - *Connecting the DSTREAM unit*, `http://infocenter.arm.com/help/topic/com.arm.doc.dui0481-/I1004916.html`.

3.7 Using an SSH connection to set up and run Gnometris on an ARM Linux target

On some targets you can use a *Secure SHell* (SSH) connection with the *Remote System Explorer* (RSE) provided with DS-5.

To set up a Linux SSH connection to an ARM Linux target and run the Gnometris application:

1. Add the Remote Systems view to the DS-5 Debug perspective:
   a. Ensure that you are in the DS-5 perspective. To change perspective either use the perspective toolbar or select **Window → Open perspective → DS-5 Debug** from the main menu.
   b. Select **Window → Show View → Other...** to open the Show View dialog box.
   c. Select the **Remote Systems** view in the **Remote Systems** group.
   d. Click **OK**.
2. In the Remote Systems view, set up a Linux connection to a remote target using SSH:
   a. Click on **Define a connection to remote system** in the Remote Systems view toolbar.
   b. In the Select Remote System Type dialog box, expand the **General** group and select **Linux**.

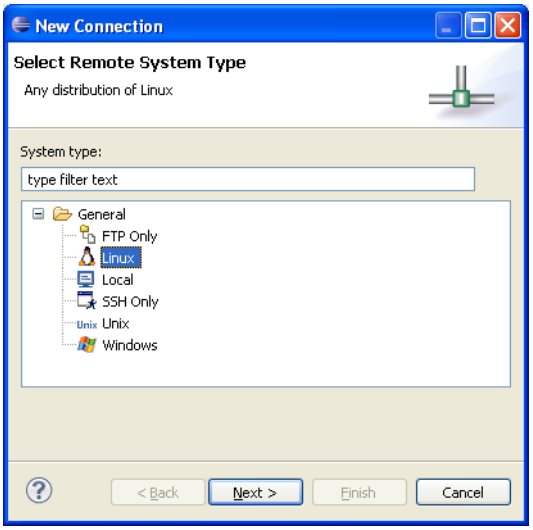

Figure 3-1 Selecting a connection type

   c. Click **Next**.
   d. In the Remote Linux System Connection, enter the remote target IP address or name in the Host name field.

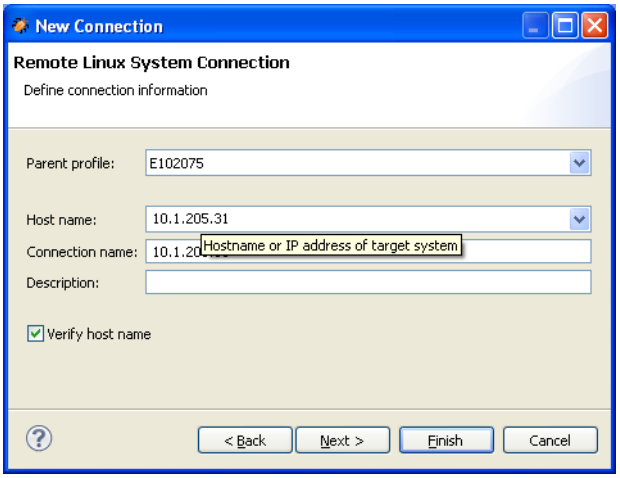

Figure 3-2 Defining the connection information

e. Click **Next**.

f. Select SSH protocol file access.

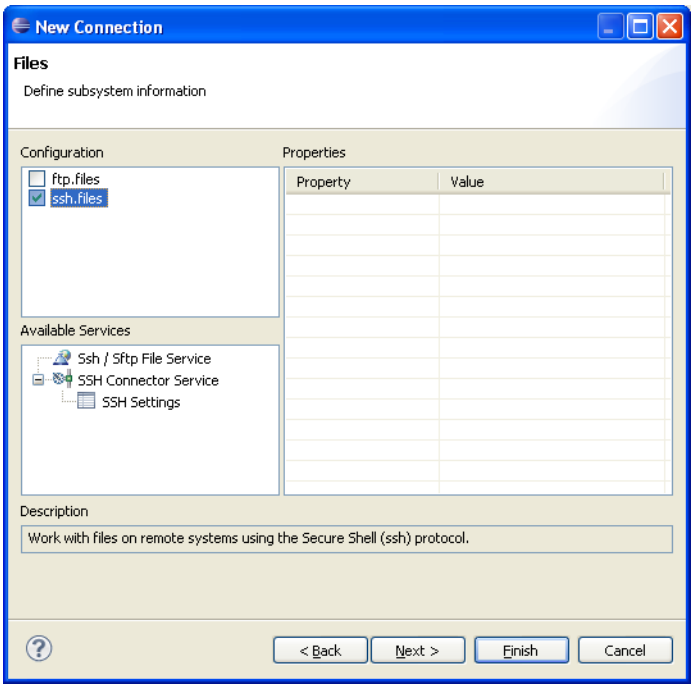

Figure 3-3 Defining the file system

g. Click **Next**.

h. Select the shell processes for Linux systems.

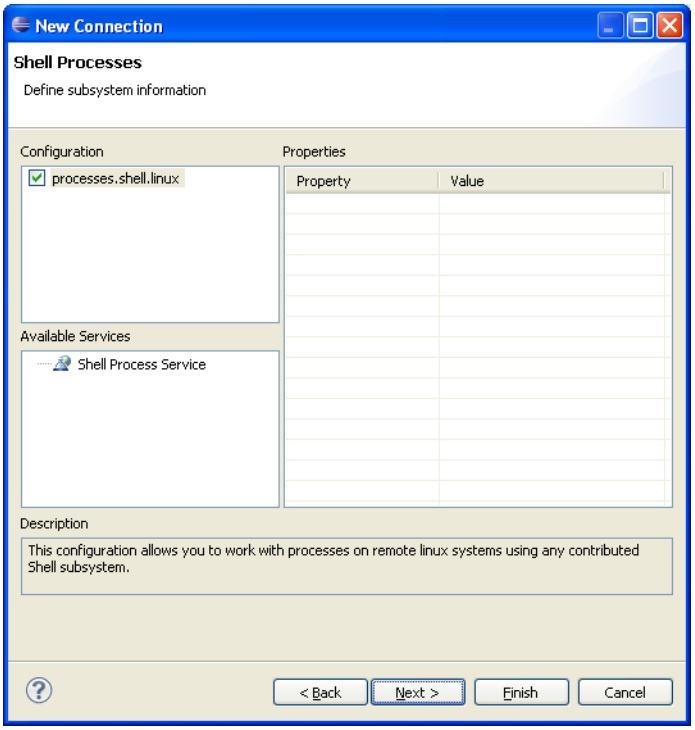

Figure 3-4 Defining the processes

i. Click **Next**.

j. Select SSH shells.

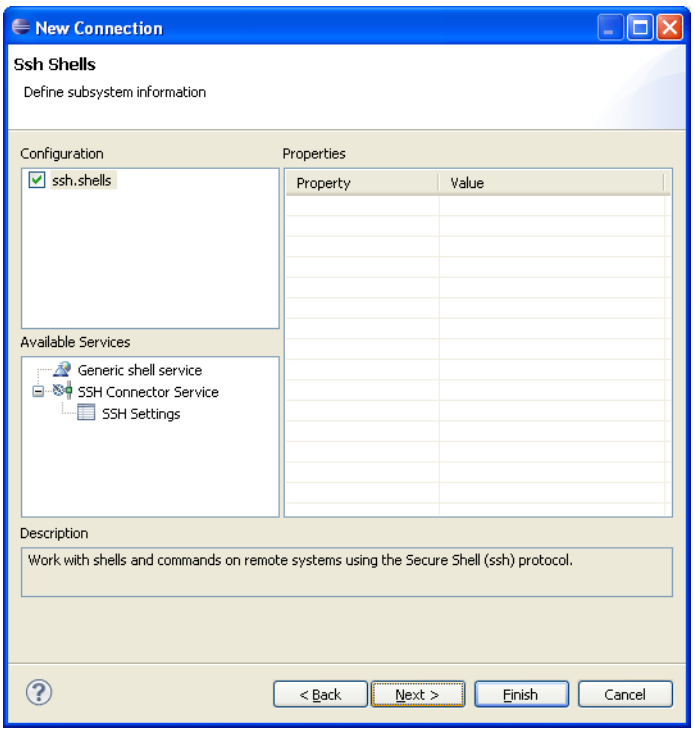

Figure 3-5 Defining the shell services

k. Click **Next**.

l. Select SSH terminals.

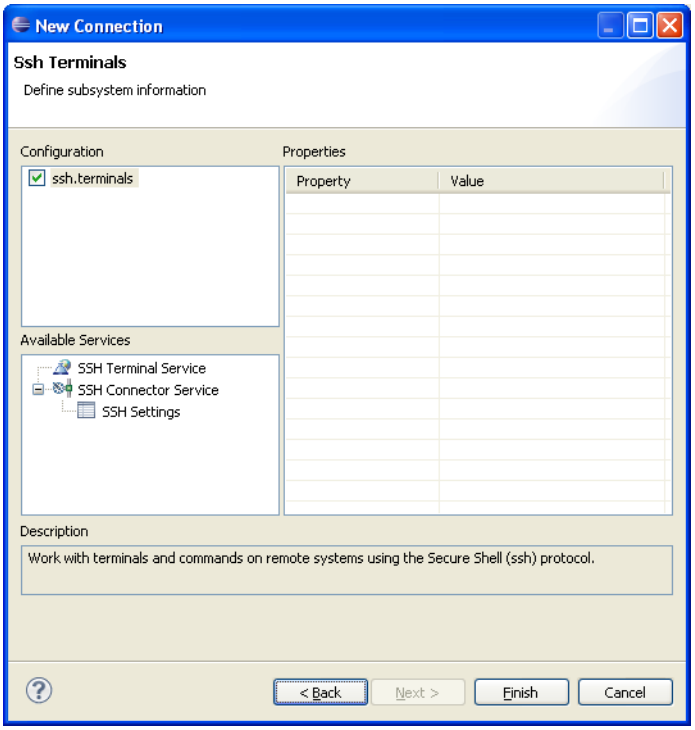

Figure 3-6 Defining the terminal services

m. Click **Finish**.

3. In the Remote Systems view:
   a. Right-click on the Linux target and select **Connect** from the context menu.
   b. In the Enter Password dialog box, enter a **User ID** and **Password** if required.
   c. Click **OK** to close the dialog box.
   d. Copy the stripped version of the Gnometris application, `gnometris`, and the `libgames-support.so` library from the local file system on to the target file system.
   e. Ensure that the files on the target have execute permissions. To do this, right-click on each file, select **Properties** from the context menu and change the checkboxes as required.

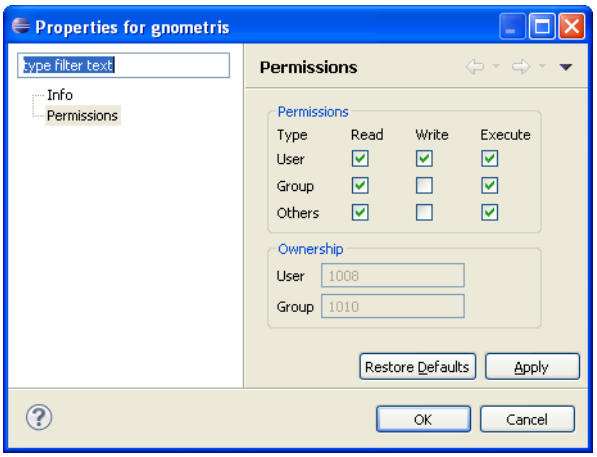

Figure 3-7 Modifying file properties from the Remote Systems view

4. Open a terminal shell that is connected to the target and launch gdbserver with the application:
   a. In the Remote Systems view, right-click on **Ssh Terminals**.
   b. Select **Launch Terminal** to open a terminal shell.
   c. In the terminal shell, navigate to the directory where you copied the `gnometris` application, then execute the following command:

```
export DISPLAY=ip:0.0
gdbserver :port gnometris
```

   where:

   ip is the IP address of the host to display the Gnometris game

   port is the connection port between `gdbserver` and the application, for example `5000`.

Note

If the target has a display that you can use, then you do not need to `export DISPLAY`.

3.7.1 See also

Tasks

- *Connecting to the Gnometris application that is already running on a ARM Linux target* on page 3-13
- *Debugging Gnometris* on page 3-16.

Reference

- *Examples* on page 4-6
- *ARM® DS-5™ Using the Debugger*:
  - *Debug Configurations - Connection tab on page 11-60*
  - *Debug Configurations - Files tab on page 11-64*
  - *Debug Configurations - Debugger tab on page 11-68*
  - *Connecting the DSTREAM unit*, `http://infocenter.arm.com/help/topic/com.arm.doc.dui0481-/I1004916.html`.

3.8 Connecting to the Gnometris application that is already running on a ARM Linux target

To connect the debugger to the Gnometris application that is already running on an ARM Linux target:

1. Select **Debug Configurations...** from the **Run** menu.
2. Select **DS-5 Debugger** from the configuration tree and then click on **New** to create a new configuration. Alternatively you can select an existing DS-5 Debugger configuration and then click on **Duplicate** from the toolbar.
3. In the Name field, enter a suitable name for the new configuration.
4. Click on the **Connection** tab to see the target and connection options.
5. In the Select target panel:
   a. Select the required platform, for example, **beagleboard.org - OMAP_3530**.
   b. Select **Connect to already running gdbserver** for the debug operation.
6. In the Connections panel, for the connection between `gdbserver` and the application:
   a. Enter the IP address of the target.
   b. Enter the port number.

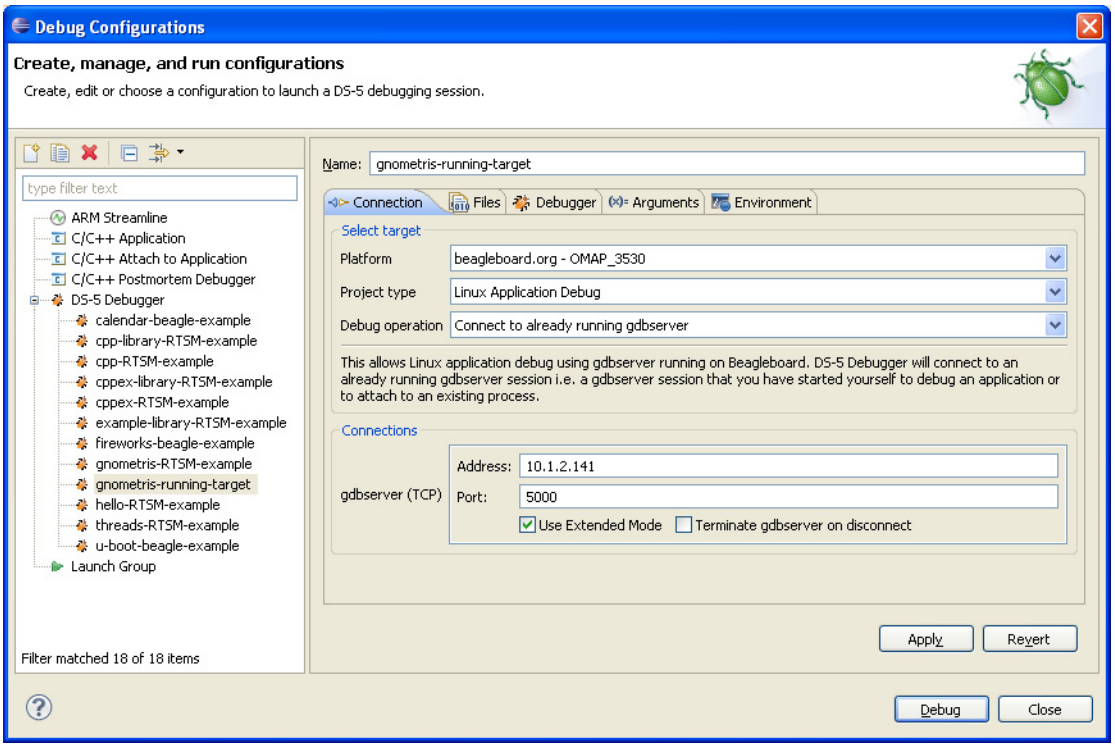

Figure 3-8 Typical connection configuration for a Beagle board

7. Click on the **Files** tab to see the file options.
8. In the Files panel:
   a. Select **Load symbols from file** and then select the application image containing debug information. For example: `H:\workspace\gnometris\gnometris`.
   b. Click **Add a new resource to the list** to add another file entry.

c. Select **Load symbols from file** and then select the shared library that is required by the Gnometris application. For example:
`H:\workspace\gnometris\libgames-support.so`.

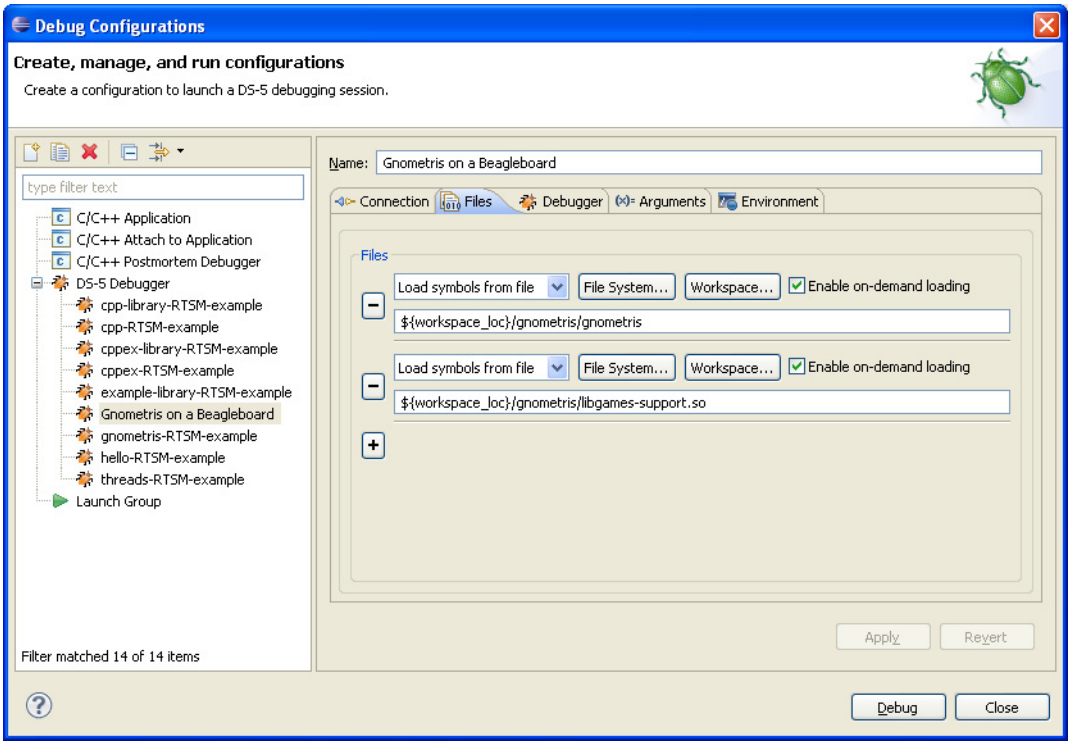

Figure 3-9 Typical file selection for a Beagle board

9. Click on the **Debugger** tab to see the debugging options for the configuration.
10. In the Run control panel:
    a. Select **Debug from symbol**.
    b. Enter **main** in the field provided.
11. In the Host working directory panel, select **Use default**.

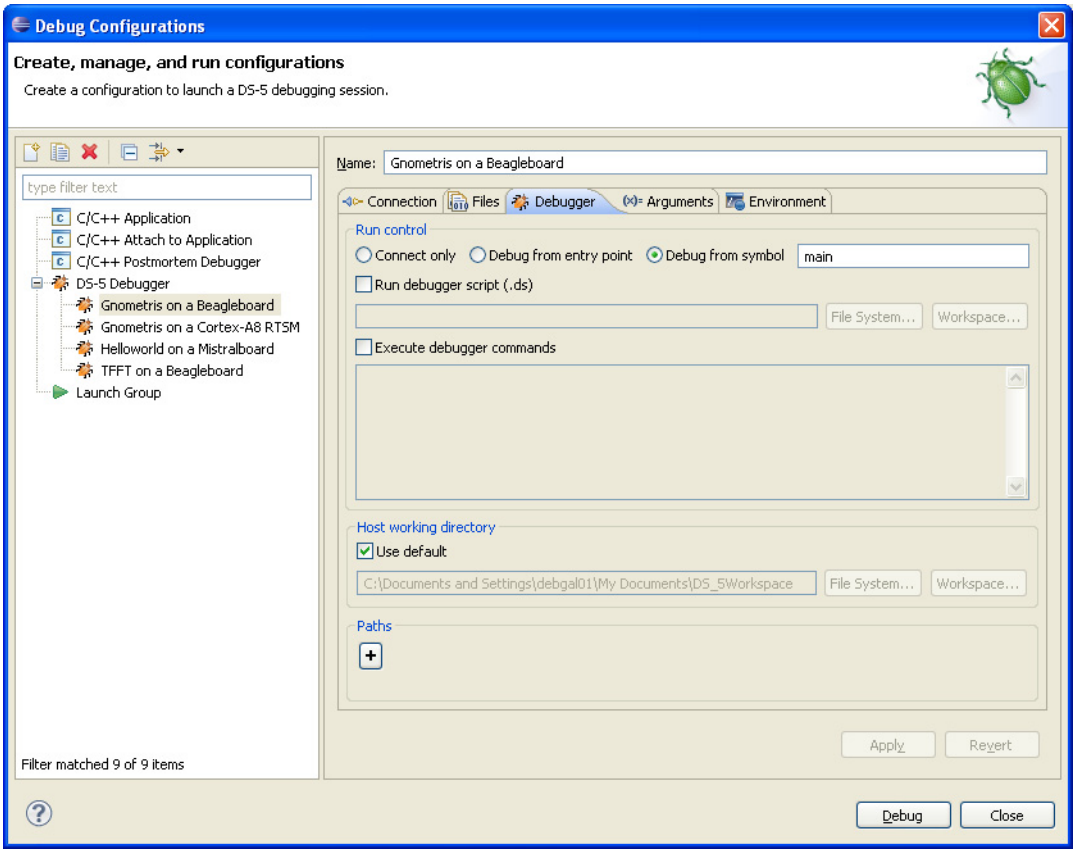

Figure 3-10 Typical debugger settings for a Beagle board

12. Click on **Debug** to start the debugger and run to the `main()` function.

13. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.

3.8.1 See also

Tasks

- *Debugging Gnometris* on page 3-16.

Reference

- *Examples* on page 4-6
- *ARM® DS-5™ Using the Debugger*:
  - *Debug Configurations - Connection tab on page 11-60*
  - *Debug Configurations - Files tab on page 11-64*
  - *Debug Configurations - Debugger tab on page 11-68*
  - *Connecting the DSTREAM unit*, `http://infocenter.arm.com/help/topic/com.arm.doc.dui0481-/I1004916.html`.

3.9 Debugging Gnometris

To debug the Gnometris application:

1. Ensure that you are connected to the target, Gnometris is running, and the debugger is waiting at the `main()` function.
2. In the Project Explorer view, open the Gnometris directory to see a list of all the source files.
3. Double-click on the file `blockops-noclutter.cpp` to open the file.
4. In the `blockops-noclutter.c` file, find the line `BlockOps::rotateBlock()`, and double click in the vertical bar on the left-hand side of the C/C++ editor to add a breakpoint. A marker is placed in the vertical bar of the editor and the Breakpoints view updates to display the new information.
5. Click on **Continue** in the Debug Control view to continue running the program.
6. Start a new Gnometris game on the target. When a block arrives, press the up cursor key to hit the breakpoint.
7. Select the Registers view to see the values of the registers.
8. Select the Disassembly view to see the disassembly instructions. You can also double click in the vertical bar on the left-hand side of this view to set breakpoints on individual instructions.
9. In the Debug Control view, click on **Step Over Source Line** to move to the next line in the source file. All the views update as you step through the source code.
10. Select the History view to see a list of all the debugger commands generated during the current debug session. You can select one or more commands and then click on **Exports the selected lines as a script** to create a script file for future use.

3.9.1 See also

Tasks

- *Importing the example projects into Eclipse* on page 3-2
- *Building the Gnometris project from Eclipse* on page 3-4
- *Building the Gnometris project from the command-line* on page 3-5
- *Loading the Gnometris application on a Real-Time System Model* on page 3-6.
- *Loading the Gnometris application on to an ARM Linux target* on page 3-7.
- *ARM® DS-5™ Using the Debugger*:
  - *Configuring a connection to a Linux target using gdbserver on page 3-5*.
- *ARM® DS-5™ Using Eclipse*:
  - *Remote Systems view* on page 6-3.

Reference

- *Examples* on page 4-6
- *ARM® DS-5™ Using the Debugger*:
  - *C/C++ editor on page 11-12*
  - *Debug Control view on page 11-18*
  - *Registers view on page 11-36*.

3.10 Debugging a loadable kernel module

You can use DS-5 to develop and debug a loadable kernel module. Loadable modules can be dynamically inserted and removed from a running kernel during development without the need to frequently recompile the kernel.

DS-5 provides an example of a simple character device driver, `modex.c` that you can compile, run, and debug on your target. Pre-built image binaries are provided for Windows users that match the Linux distribution project provided by DS-5. Alternatively, see the `readme.html` provided with the `kernel_module` example for more information on how to compile the kernel and the module.

3.10.1 Prerequisites

Before you can debug the module you must ensure that you:

- Unpack kernel source code and compile the kernel against exactly the same kernel version as the target
- Compile the loadable module against exactly the same kernel version as the target.

Note

Ensure that you compile both images with debug information. The debugger requires run-time information from both images when debugging the module.

3.10.2 Procedure

To debug the loadable module, `modex.c`:

1. Connect the debugger to the target. The device driver example provides a preconfigured launch file:
   a. Select **Debug Configurations...** from the **Run** menu.
   b. Expand the **DS-5 Debugger** the configuration tree.
   c. Select the `module-beagle-example` entry.
   d. The **Connection** tab contains most of the connection settings with the exception of the Debug Hardware Address field. Enter the IP address or name for the connection between the debugger and the debug hardware agent.

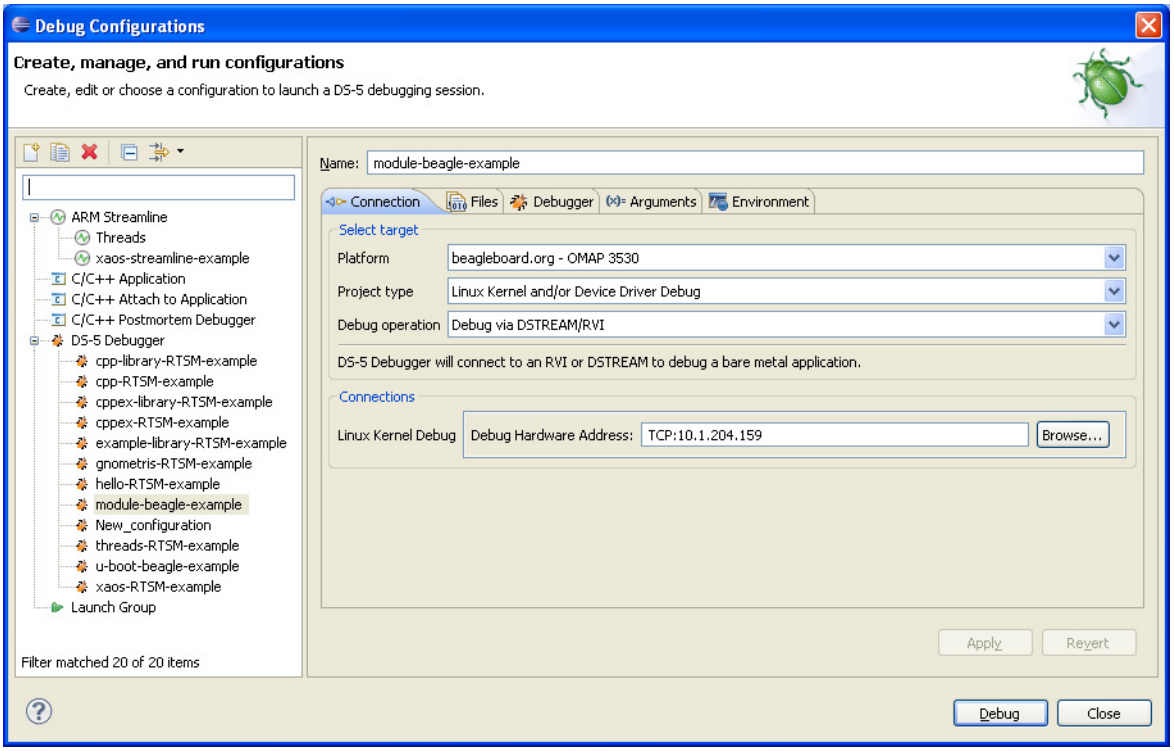

Figure 3-11 Typical connection for a Linux kernel module configuration

e. The **Files** tab contains the debugger settings to load debug information for the Linux kernel and the module. For this example, ignore the Application on host to download field and select both the kernel image and the module image as shown in the following figure.

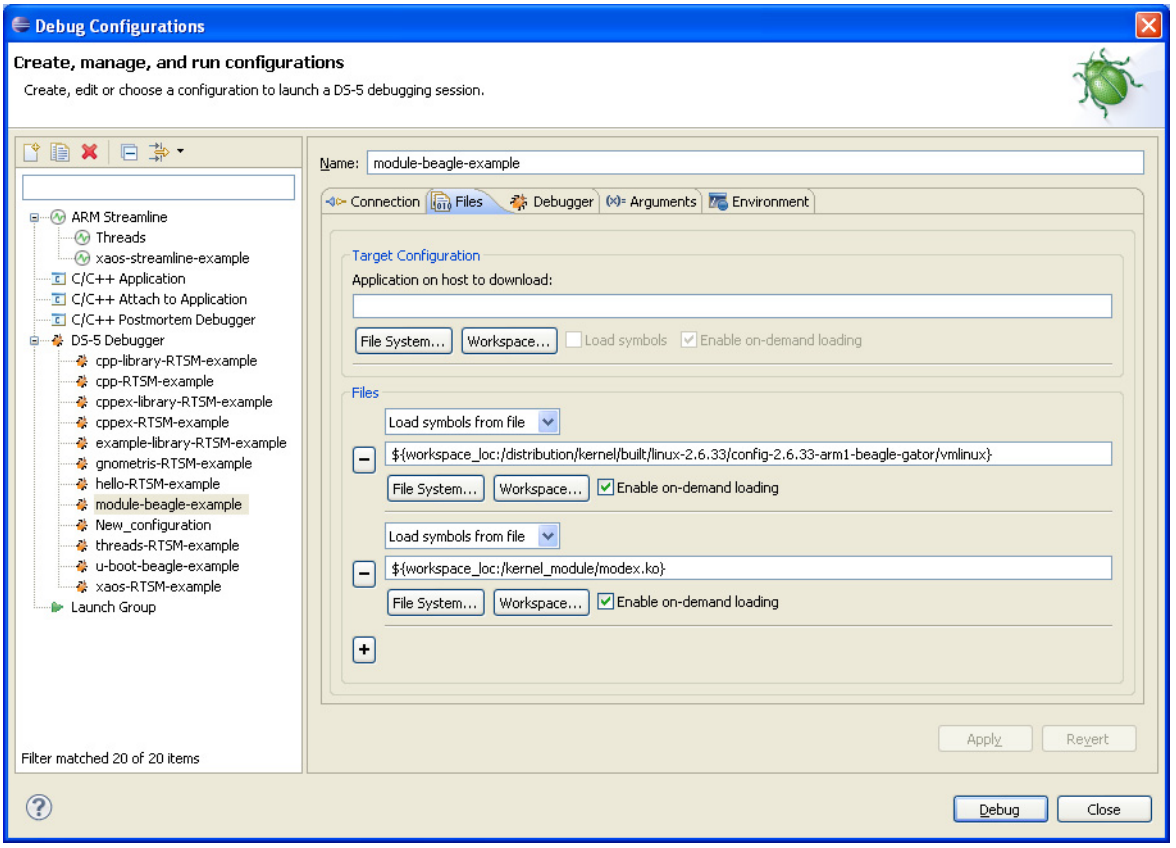

Figure 3-12 Typical file selection for a Linux kernel module configuration

f. In the **Debugger** tab, select **Connect only** in the Run control panel.

g. Click on **Debug** to connect the debugger to the target.

2. Configure and connect a terminal shell to the target. You can use the *Remote System Explorer* (RSE) provided with DS-5.

3. Using RSE, copy the compiled module to the target:

   a. Navigate to the `.../linux_system/kernel_module/stripped` directory on the host workstation.

   b. Drag and drop the module, `modex.ko` to a writeable directory on the target.

4. In the terminal shell, insert the module:

   a. Navigate to the location of the module.

   b. Execute the following command:

   ```
   insmod modex.ko
   ```

   The Modules view updates to display details of the loaded module.

5. To debug the module, set breakpoints, run, and step as required.

6. To modify the module source code:

   a. Remove the module. For example:

   ```
   rmmod modex
   ```

   b. Recompile the module.

   c. Repeat steps 3, 4 and 5 as required.

Note

When you insert and remove a module, the debugger stops the target and automatically resolves memory locations for debug information and existing breakpoints. This means that you do not have to stop the debugger and reconnect when you recompile the source code.

Useful terminal shell commands:

`lsmod`	Displays information about all the loaded modules.
`insmod`	Inserts a loadable module.
`rmmod`	Removes a module.

Useful DS-5 Debugger commands:

`info os-modules`	Displays a list of OS modules loaded after connection.
`info os-log`	Displays the contents of the OS log buffer.
`info os-version`	Displays the version of the OS.
`info processes`	Displays a list of processes showing ID, current state and related stack frame information.
`set os-log-capture`	Controls the capturing and printing of *Operating System* (OS) logging messages to the console.

OS modules loaded after connection are displayed in the Modules view.

3.10.3 See also

Tasks

- *ARM® DS-5™ Using the Debugger*:
  - *Configuring a connection to a Linux Kernel on page 3-7*
  - Chapter 4 *Controlling execution*
  - Chapter 5 *Examining the target*.

Concepts

- *ARM® DS-5™ Using the Debugger*:
  - *About debugging a Linux kernel on page 6-10*
  - *About debugging Linux kernel modules on page 6-12*
  - *ARM Linux problems and solutions on page 12-2*
  - *Target connection problems and solutions on page 12-4*.

Reference

- *Examples* on page 4-6
- *ARM® DS-5™ Using the Debugger*:
  - *Breakpoints view on page 11-8*
  - *Commands view on page 11-15*
  - *Debug Control view on page 11-18*
  - *Modules view on page 11-34*.
- *ARM® DS-5™ Debugger Command Reference*:
  - *info os-log* on page 2-90

  - — *info os-modules* on page 2-91
  - — *info os-version* on page 2-92
  - — *info processes* on page 2-93
  - — *set os* on page 2-149
  - — *show os* on page 2-176.
- *ARM® DS-5™ Using Eclipse*:
  - — *Terminals view* on page 6-6.
- *Connecting the DSTREAM unit*, `http://infocenter.arm.com/help/topic/com.arm.doc.dui0481-/I1004916.html`.

3.11 Performance analysis of threads application running on ARM Linux

ARM Streamline is a graphical performance analysis tool. It provides timeline and analysis reports that highlight problem areas at system, process, and thread level, in addition to hot spots in the applications.

3.11.1 Prerequisites

Before capturing the analysis data, ensure that:

1. You obtain the IP address of the target. You can use the `ifconfig` application in a Linux console. The IP address is denoted by the **inet addr**.
2. The ARM Linux Kernel is configured for Streamline.
3. The threads application is copied to the target.
4. The gator daemon is running on the target.

3.11.2 Procedure

To capture the data:

1. Launch Eclipse.
2. Launch a terminal shell and connect it to the target. You can use the terminal shell provided with *Remote System Explorer* (RSE).
3. In the terminal shell, navigate to the directory where you copied the `threads` application.
4. Ensure that you are in the C/C++ Perspective.
5. Create a target connection:
   a. Select the **Change capture options...** toolbar icon in the Streamline Capture Data view.
   b. In the Name field, enter a suitable name for the new configuration.
   c. In the Connection panel, enter the IP address or name and the associated port number for the connection between the host workstation and the target.
   d. In the Capture panel, accept the default settings or customize as required.
   e. Click on **Add Program...** or **Add program from Workspace...** in the Program Images panel to open a dialog box where you can select the application image.
   f. Navigate to the threads application and click on **Open** or **OK** to close the dialog box.
   g. Click on **Apply** to save the settings.
   h. To start capturing the data, click on the **Start capture** toolbar icon in the Streamline Capture Data view.
6. In the terminal shell, execute the following command to run the threads application:
   ```
   ./threads
   ```
7. After you have completed running the threads application, return to the C/C++ Perspective in Eclipse.
8. Click on the report in the Streamline Capture Data view to analyze the graphical data.

Figure 3-13 Streamline Capture Data file

A Streamline Analysis Data file is generated automatically when you stop capturing the data or you can double-click on an existing analysis file to view it in the Editor.

Figure 3-14 Streamline Analysis Data file

3.11.3 See also

Concepts

- *About ARM Streamline Performance Analyzer* on page 2-8.

Reference

- *Documentation* on page 4-5
- *Examples* on page 4-6
- *ARM® DS-5™ Using Eclipse*:
  - *Terminals view* on page 6-6
- *ARM® DS-5™ Using ARM Streamline*, `http://infocenter.arm.com/help/topic/com.arm.doc.dui0482-/index.html`.

3.12 Debugging Android native C/C++ applications and libraries

This tutorial describes how to debug the `hello-neon` application provided with the Android *Native Development Kit* (NDK). It uses the Android SDK Platform 2.2 and the Android emulator as the target.

Note

It does not describe how to install any of the Android tools. See the *Android Developers* website for more information.

3.12.1 Prerequisites

Before you can debug an Android package containing native C/C++ code you must:

1. Download and install the Android *Software Development Kit* (SDK). This enables you to build Java applications together with any native C/C++ code into an Android package with a `.apk` file extension.
2. Download and install the Android NDK. This is a companion tool to the Android SDK that enables you to build performance-critical parts of your applications in native code such as C and C++ languages.

   Note

   On Windows, you must also download and install `cygwin`, including the `make` package so that you can run the scripts inside the Android NDK.

3. Update the version of `gdbserver` in the relevant Android NDK toolchain directory by copying the Android version provided with DS-5 *`arm_directory`*`\gdbserver\...\android`. This tutorial uses the `...\toolchains\arm-eabi-4.4.0\prebuilt` directory.
4. Set up the Eclipse plug-in for Android:
   a. Launch Eclipse.
   b. Install the *Android Development Tools* (ADT) Eclipse plug-ins. For example, from the following site: `http://dl-ssl.google.com/android/eclipse`.
   c. Select **Window** → **Preferences** → **Android** and click on **Browse...** to set the location of the Android SDK.
   d. Open the Android SDK and AVD Manager dialog box by selecting **Window** → **Android SDK and AVD Manager**.
   e. Expand the **Available packages** group and add SDK platforms as required. For example, Android SDK Platform Android 2.2.
   f. Create a new *Android Virtual Device* (AVD).
5. Edit the Android NDK script file, `ndk-gdb` to debug using DS-5. The Android NDK contains a script file to run `gdbserver` and the application on the target before launching the debugger. By default, the script file is not set up to debug using DS-5. To change this behavior you must comment out the last line as shown:

   ```
   #$GDBCLIENT -x `native_path $GDBSETUP`
   ```

3.12.2 Procedure

To debug the application:

1. Build the `hello-neon` source files with debug information using the scripts provided by the Android NDK. This tutorial uses the `...\toolchains\arm-eabi-4.4.0\prebuilt` directory. For example:

   ```
   ./ndk-build -C samples/hello-neon NDK_TOOLCHAIN=arm-eabi-4.4.0 NDK_DEBUG=1
   ```

2. Launch Eclipse.

3. Create a new Android project:
   a. Select **File → New → Project...**
   b. Expand the Android group and select **Android Project**.
   c. Click **Next**.
   d. Enter a suitable project name. For example, `HelloNeon`.
   e. Select **Create project from your existing source** and locate the `hello-neon` folder.
   f. Leave the **Use default location** option selected so that the project is created in the default directory shown. Alternatively, deselect this option and browse to your preferred project directory.
   g. Select the required Build Target. For example, **Android 2.2**.
   h. Enter a suitable Application name. For example, `Hello, Neon`.
   i. Enter a suitable Package name. For example, `com.example.neon`.
   j. Enter a suitable Activity name. For example, `HelloNeon`.
   k. Click **Finish**.

4. Ensure that the application builds with debug information. You can do this by:
   a. Open the `AndroidManifest.xml` file.
   b. Click on the **Application** tab.
   c. Select **true** in the Debuggable field.
   d. Save the changes and close the file.

5. Clean and rebuild the Android project.

6. If the application is already installed on the target you must uninstall it. For example:

   ```
   path\adb uninstall com.example.neon
   ```

7. Install the application. For example:

   ```
   path\adb install samples/hello-neon/bin/HelloNeon.apk
   ```

8. Run the `ndk-gdb` script to start the application and connect `gdbserver`. For example:

   ```
   ./ndk-gdb --project=samples/hello-neon --verbose --port=5000 --start --force
   --adb=adb
   ```

   See the Android NDK documentation for more information on using the script file and the command-line options.

9. Connect DS-5 to the application using a `gdbserver` TCP connection:
   a. Select **Debug Configurations...** from the **Run** menu.
   b. Select **DS-5 Debugger** from the configuration tree and then click on **New** to create a new configuration. Alternatively you can select an existing DS-5 Debugger configuration and then click on **Duplicate** from the toolbar.
   c. In the Name field, enter a suitable name for the new configuration.

d. Click on the **Connection** tab and configure a DS-5 Debugger target connection as shown in the following figure.

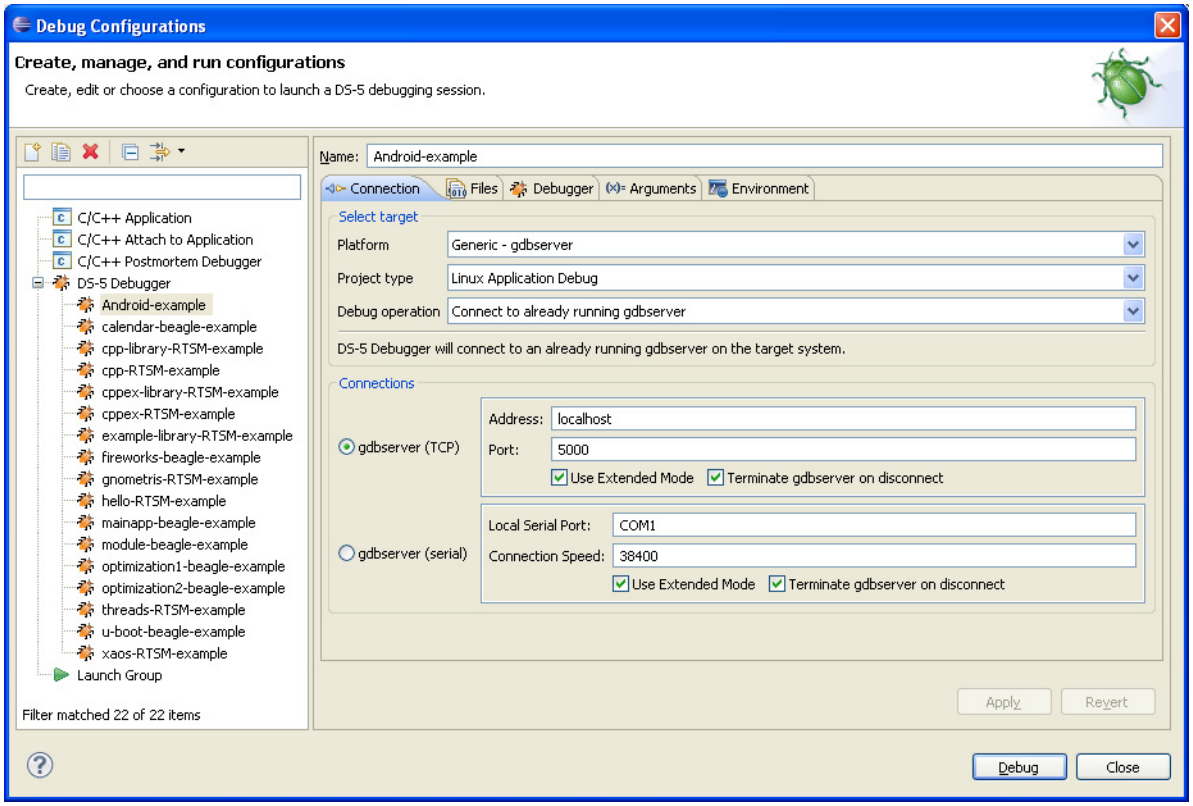

Figure 3-15 Typical connection configuration for an Android application

e. Click on the **Files** tab and select the `app_process` object file.

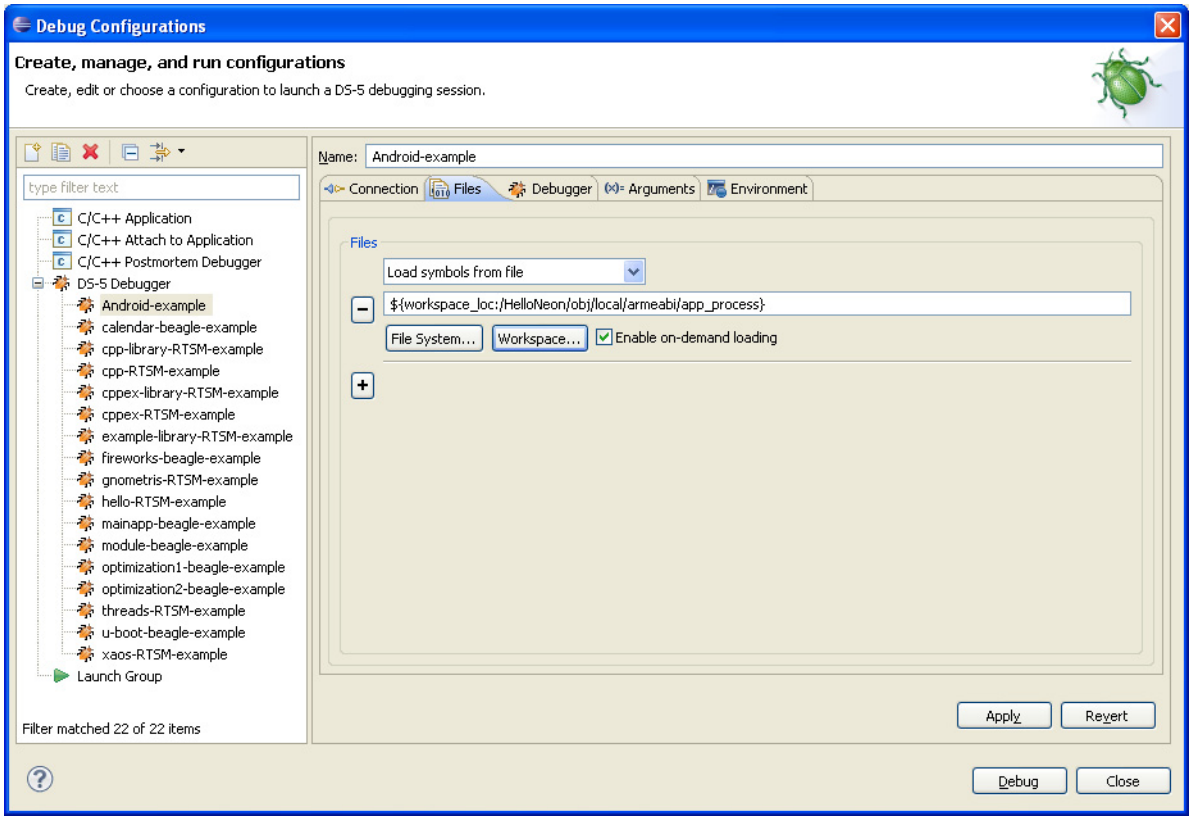

Figure 3-16 Typical file selection for an Android application

f. Click on the **Debugger** tab and select **Connect only** in the Run Control panel.

g. Select **Execute debugger commands** and enter `sharedlibrary` in the associated text box to load debug information from all shared libraries into the debugger.

h. In the Paths panel, specify the shared library search directory on the host that the debugger uses when it displays source code.

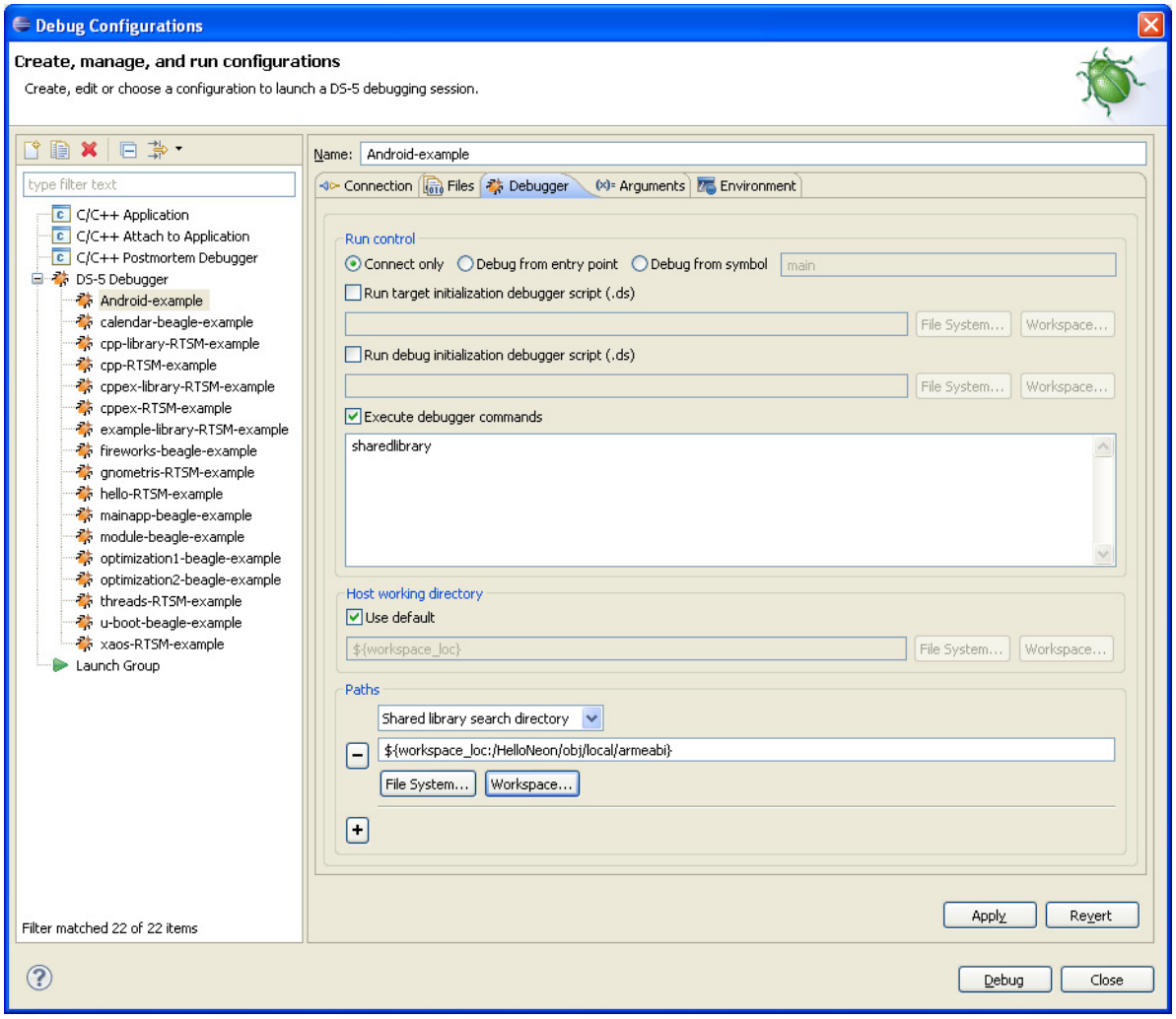

Figure 3-17 Typical debugger connection settings for an Android application

i. Click on **Debug** to connect to the target.

10. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.

11. To debug the application, set breakpoints, run, and step as required.

3.12.3 See also

Reference

- *ARM® DS-5™ Using Eclipse:*
  - *Installing new features* on page 3-40
  - *Adding a new source file to your project* on page 4-16
  - *Linked resources* on page 3-11.

Other information

- *DS-5 Knowledge Articles*, `http://infocenter.arm.com/help/topic/com.arm.doc.faqs/kiXXwMK1Sxk7vf.html`
- *Android Developers*, `http://developer.android.com`.

3.13 Managing DS-5 licenses

You can manage DS-5 licenses by selecting **ARM License Manager...** from the **Help** menu within Eclipse.

Installed licenses are display in the ARM License Manager dialog box.

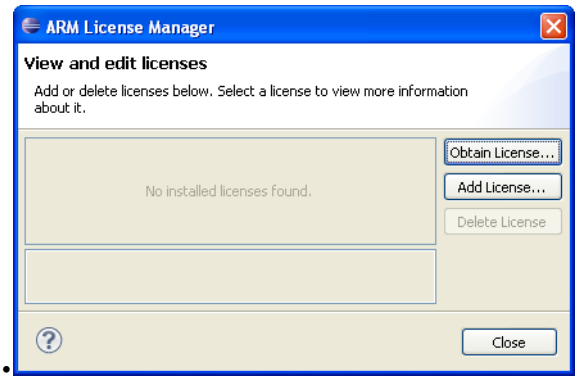

Figure 3-18 View and edit licenses

- Click on **Obtain License...** to request a new license and follow the instructions in the dialog box.

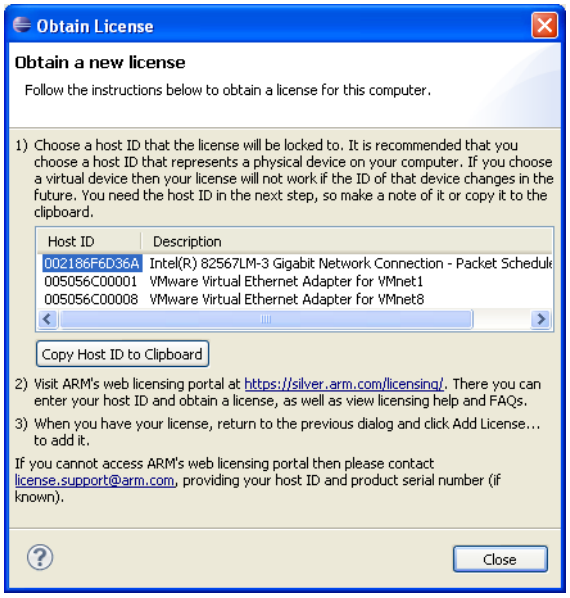

Figure 3-19 Obtain a new license

- Click on **Add License...** to install a new license. License files are copied into the `%APPDATA%\ARM\DS-5\licenses` folder for Windows and `$HOME/.ds-5/licenses` folder for Linux. Server licenses can be entered separately in the host and port fields or you can paste the complete string *`port@host`* in the Host field.

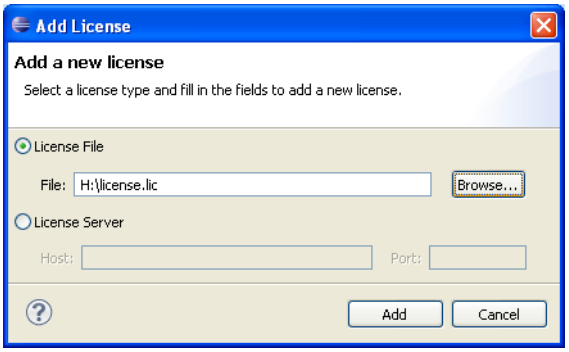

Figure 3-20 Add a new license

- Click on **Delete License** to uninstall the license and remove the file from the DS-5 license folder.

3.13.1 See also

Reference

- *Licensing and product updates* on page 4-4
- *ARM® DS-5™ License Management Guide*, `http://infocenter.arm.com/help/topic/com.arm.doc.dui0577-/index.html`.

Other information

- *ARM Self-Service Portal*, `http://silver.arm.com/`.

Chapter 4
ARM DS-5 installation and examples

The following topics describe the installation and licensing requirements. It also includes information on the documentation and examples provided with ARM® DS-5™.

Reference

4.1 System requirements

To install and use DS-5, you must have a minimum specification of computer with a dual core 2GHz processor (or equivalent) and 2GB of RAM. 4GB or more of RAM is recommended to improve performance when debugging large images, using models with large simulated memory maps, or when using ARM Streamline™ Performance Analyzer.

A full installation requires approximately 1.5 GB of hard disk space.

4.1.1 Supported platforms

DS-5 is supported (except where specified) on 32-bit and 64-bit versions of the following platforms (and service packs):

- Windows XP Professional service pack 3 (32-bit only)
- Windows 7 Professional
- Windows 7 Enterprise
- Windows Server 2003 (ARM Compiler only)
- Windows Server 2008 (ARM Compiler only)
- Red Hat Enterprise Linux 5 Desktop and Workstation option, Standard.

4.1.2 DS-5 requirements

Android and ARM Linux application debug require `gdbserver` to be available on your target. The recommended version of `gdbserver` is 6.8. `gdbserver` 7.0 executables built for ARMv4T™, ARMv5T™, and Thumb®-2 architectures are provided with DS-5 in the *`install_directory`*`\arm` directory. DS-5 Debugger is unable to provide reliable multi-threaded debug support with gdbserver versions prior to 6.8.

DS-5 Debugger supports debugging ARM Linux kernel versions 2.6.28 to 2.6.36. Other kernel versions might work, but have not been tested. The minimum ARM Linux kernel version for use with ARM Streamline Performance Analyzer is 2.6.32. Application debug on *Symmetric Multi-Processing* (SMP) systems requires ARM Linux kernel version 2.6.36 or later.

ARM Linux kernel and bare-metal debugging require the use of a DSTREAM or RVI unit with the latest firmware for DS-5 target connections. Use the debug hardware configuration utilities to check the firmware version that is currently installed and update it if necessary. Updated firmware is available in the *`install_directory`*`/sw/debughw/firmware` directory.

4.1.3 See also

Reference

- *About Debug hardware configuration utilities* on page 2-9
- *Installation directories* on page 4-3
- *Licensing and product updates* on page 4-4
- *ARM® DSTREAM™ Setting up the Hardware*, http://infocenter.arm.com/help/topic/com.arm.doc.dui0481-
- *ARM® RVT™ and RVT™ Setting up the Hardware*, http://infocenter.arm.com/help/topic/dui0515-
- *DS-5 Knowledge Articles*, http://infocenter.arm.com/help/topic/com.arm.doc.faqs/kiXXwMK1Sxk7vf.html.

4.2 Installation directories

Various directories are installed with DS-5 that contain example code and documentation. The DS-5 documentation refers to these directories as required.

The main installation, examples, and documentation directories are identified in the following table. The *install_directory* shown is the default installation directory. If you specify a different installation directory, then the path names are relative to your chosen directory.

Table 4-1 DS-5 default directories

Directory	Windows	Linux
install_directory	C:\Program Files\DS-5	~/ds-5
arm_directory	*install_directory*\arm\...	*install_directory*/arm/...
examples_directory	*install_directory*\examples\...	*install_directory*/examples/...
tools_directory	*install_directory*\bin\...	*install_directory*/bin/...
documents_directory	*install_directory*\documents\...	*install_directory*/documents/...

4.2.1 See also

Reference

- *Documentation* on page 4-5
- *Examples* on page 4-6.

4.3 Licensing and product updates

DS-5 is a licensed product that uses the FLEX*net* license management software to enable features corresponding to specific editions.

Table 4-2 DS-5 Editions

	Application Edition	Linux Edition	Professional Edition
Eclipse for DS-5	X	X	X
GNU Compilation Tools	X	X	X
Linux application debug	X	X	X
ARM Streamline Performance Analyzer	X	X	X
Cortex™-A8 real-time system model	X	X	X
Kernel space debug and trace		X	X
Bare-metal debug and trace		X	X
ARM Compiler toolchain			X

To request a license or to access the latest DS-5 product information and updates, go to the ARM Self-Service Portal.

You can access the license management software by selecting **ARM License Manager...** from the **Help** menu in Eclipse for DS-5.

4.3.1 See also

Tasks

- *Managing DS-5 licenses* on page 3-29.

Reference

- *ARM® DS-5™ License Management Guide*, http://infocenter.arm.com/help/topic/com.arm.doc.dui0577-/index.html.

Other information

- *ARM Self-Service Portal*, http://silver.arm.com/.

4.4 Documentation

The DS-5 documentation suite comprises:

- *ARM® DS-5™ Getting Started with DS-5* (this document)
- *ARM® DS-5™ Using the Debugger*
- *ARM® DS-5™ Debugger Command Reference*
- *ARM® DS-5™ Using Eclipse*
- *ARM® DSTREAM™ Setting Up the Hardware*
- *ARM® DSTREAM™ System and Interface Design Reference*
- *ARM® RVI™ and RVT™ Setting Up the Harware*
- *ARM® RVI™ and RVT™ System and Interface Design Reference*
- *ARM® DSTREAM™ and RVT™ Using the Debug Hardware Configuration Utilities*
- *ARM® Streamline™ Performance Analyzer Using ARM Streamline*

To access the DS-5 documentation:

1. Launch Eclipse:
   - On Windows, select **Start → All Programs → ARM DS-5 → Eclipse for DS-5**.
   - On Linux, enter `eclipse` in the Unix bash shell.
2. Select **Help Contents** from the **Help** menu.

Documentation on using the examples is available in *`examples_directory`*`\docs`.

Documentation on using the GNU compilation tools is available in *`documents_directory`*`\gcc`.

4.4.1 See also

Reference

- *Installation directories* on page 4-3
- *Examples* on page 4-6
- *Documentation on the ARM website*, http://infocenter.arm.com/help/topic/com.arm.doc.subset.swdev.ds5.

4.5 Examples

DS-5 provides a selection of examples to help you get started:

- Bare-metal software development examples that illustrate `armcc` managed builder, bare-metal debug, performance optimization, and measurement techniques. The files are located in the archive file, *`examples_directory`*`\Bare-metal_examples.zip`.
- Bare-metal example projects for supported boards that demonstrate board connection and basic debug into on-chip RAM. The files are located in the archive file, `examples_directory\Bare-metal_boards_examples.zip`.
- ARM Linux examples that illustrate build, debug, and performance analysis of simple C/C++ console applications, shared libraries, and multi-threaded applications. These examples run on a *Real-Time System Model* (RTSM) that is preconfigured to boot ARM Linux. The files are located in the archive file, *`examples_directory`*`\Linux_examples.zip`.
- Optional packages with source files, libraries, and prebuilt images for running the examples. These can be downloaded from the **DS-5 Downloads** page on the ARM website or from the DS-5 installation media.
  - Linux distribution project with header files and libraries for the purpose of rebuilding the ARM Linux examples. The files are located in the archive file, *`examples_directory`*`\Linux_distribution_example.zip`.
  - Linux SD card image for the BeagleBoard configured for DS-5. The files are located in the archive file, *`examples_directory`*`\beagle.zip`.
  - Linux SD card image for the BeagleBoard-xM configured for DS-5. The files are located in the archive file, *`examples_directory`*`\beaglexm.zip`.

You can extract these examples to a working directory and build them from the command-line, or you can import them into Eclipse using the import wizard. All examples provided with DS-5 contain a preconfigured Eclipse launch script that enables you to easily load and debug example code on a target.

Each example provides instructions on how to build, run and debug the example code. You can access the instructions from the main index, *`examples_directory`*`\docs\index.html`.

4.5.1 See also

Tasks

- *Importing the example projects into Eclipse* on page 3-2

Concepts

- *About Real-Time System Models* on page 2-5.

Reference

- *Documentation* on page 4-5
- *Installation directories* on page 4-3
- *Using Eclipse*:
  - *Using the welcome screen* on page 3-4.

Other information

- *ARM Development Studio 5 (DS-5™)*, http://www.arm.com/products/tools/software-tools/ds-5.