

ARM[®] DS-5[™]

Version 5.12

Getting Started with DS-5

The ARM logo, consisting of the letters "ARM" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right.

ARM DS-5

Getting Started with DS-5

Copyright © 2010-2012 ARM. All rights reserved.

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
September 2011	G	Non-Confidential	Update for DS-5 version 5.7
November 2011	H	Non-Confidential	Update for DS-5 version 5.8
February 2012	I	Non-Confidential	Update for DS-5 version 5.9
May 2012	J	Non-Confidential	Update for DS-5 version 5.10
July 2012	K	Non-Confidential	Update for DS-5 version 5.11
October 2012	L	Non-Confidential	Update for DS-5 version 5.12

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

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

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

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

Where the term ARM is used it means “ARM or any of its subsidiaries as appropriate”.

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	
Chapter 2	ARM DS-5 product overview	
2.1	About DS-5	2-2
2.2	About Eclipse for DS-5	2-3
2.3	About DS-5 Debugger	2-4
2.4	About Real-Time System Models	2-5
2.5	About ARM Compiler	2-6
2.6	About GNU Compilation Tools	2-7
2.7	About ARM Streamline Performance Analyzer	2-8
2.8	About Debug hardware configuration utilities	2-9
Chapter 3	ARM DS-5 tutorials	
3.1	Installing DS-5 into a custom Eclipse environment	3-2
3.2	Importing the example projects into Eclipse	3-4
3.3	Building the Gnometris project from Eclipse	3-5
3.4	Building the Gnometris project from the command-line	3-6
3.5	Loading the Gnometris application on a Real-Time System Model	3-7
3.6	Loading the Gnometris application on to an ARM Linux target	3-8
3.7	Using an RSE connection to work with an ARM Linux target	3-9
3.8	Connecting to the Gnometris application that is already running on a ARM Linux target 3-15	
3.9	Debugging Gnometris	3-18
3.10	Debugging a loadable kernel module	3-19
3.11	Performance analysis of threads application running on ARM Linux	3-24
3.12	Setting up the Android tools for use with DS-5	3-26
3.13	Loading the hello-neon application on to an Android target	3-28
3.14	Connecting to an application that is already running on an Android target	3-32
3.15	Managing DS-5 licenses	3-36

	3.16	Changing the Toolkit	3-38
Chapter 4		ARM DS-5 installation and examples	
	4.1	System requirements	4-2
	4.2	Installation directories	4-4
	4.3	Licensing and product updates	4-5
	4.4	Documentation	4-6
	4.5	Examples	4-8

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.

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

monospace *italic*

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

`monospace` **bold**

Denotes language keywords when used outside example code.

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 0478L
- 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

- [About DS-5 on page 2-2](#)
- [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.](#)

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 utilities 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 4-5](#)
- [Documentation on page 4-6](#)
- [Examples on page 4-8.](#)

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 enable 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™ Getting Started with Eclipse:*
 - [Chapter 2 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 synchronized 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 comes pre-supplied with Cortex™-A8 and Cortex-A9 RTSMs. DS-5 can connect to a variety of ARM and third-party RTSMs too.

2.4.1 See also

Reference

- *Real-Time System Model Reference*,
<http://infocenter.arm.com/help/topic/com.arm.doc.dui0575-/index.html>

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

- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Creating a new C or C++ project on page 3-4.](#)

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-linux-gnueabi-hf-ar	GNU librarian
arm-linux-gnueabi-hf-as	GNU assembler
arm-linux-gnueabi-hf-gcc	GNU c compiler
arm-linux-gnueabi-hf-g++	GNU C++ compiler
arm-linux-gnueabi-hf-ld	GNU linker

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

2.6.1 See also

Tasks

- [Building the Gnometriz project from Eclipse on page 3-5](#)
- [Building the Gnometriz project from the command-line on page 3-6](#)
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Creating a new C or C++ project on page 3-4.](#)

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 3-24.](#)

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® DS-5™ 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

- *Installing DS-5 into a custom Eclipse environment on page 3-2*
- *Importing the example projects into Eclipse on page 3-4*
- *Building the Gnetris project from Eclipse on page 3-5*
- *Building the Gnetris project from the command-line on page 3-6*
- *Loading the Gnetris application on a Real-Time System Model on page 3-7*
- *Loading the Gnetris application on to an ARM Linux target on page 3-8*
 - *Using an RSE connection to work with an ARM Linux target on page 3-9*
 - *Connecting to the Gnetris application that is already running on a ARM Linux target on page 3-15.*
- *Debugging Gnetris on page 3-18*
- *Debugging a loadable kernel module on page 3-19*
- *Performance analysis of threads application running on ARM Linux on page 3-24*
- *Setting up the Android tools for use with DS-5 on page 3-26*
- *Loading the hello-neon application on to an Android target on page 3-28*
- *Connecting to an application that is already running on an Android target on page 3-32*
- *Managing DS-5 licenses on page 3-36*
- *Changing the Toolkit on page 3-38.*

3.1 Installing DS-5 into a custom Eclipse environment

DS-5 provides an update-site archive file for use when installing DS-5 into a custom Eclipse environment.

3.1.1 Prerequisites

The following Eclipse packages must be installed before you can use DS-5:

- 32-bit version of J2SE 6.0
- 32-bit version of Eclipse Indigo 3.7
- *C/C++ Development Tooling* (CDT) v8.0

Note

The *Eclipse IDE for C/C++ Developers* package includes CDT.

- *Remote System Explorer* (RSE) 3.3

The following optional Eclipse packages are required for Python development:

- *Java Development Toolkit* (JDT) 3.7
- *Python Development* (PyDev) 1.6.

3.1.2 Procedure

To install DS-5 into a custom Eclipse environment:

1. Launch your custom Eclipse.
2. Select **Install New Software...** from the **Help** menu.
3. Click **Add...** to add a new update site.
4. Enter a suitable name for the update site, for example: DS-5 Update Site.
5. Enter the relevant website address for the DS-5 update site. Alternatively if you have a full DS-5 installation then you can locate the archive file in the following directory:
install_directory\DS-5\sw\eclipse\update-site.zip.
6. Click on **Open**.
7. Click on **OK**.
8. Review the list of plug-ins that you are about to install and select **ARM DS-5**.
9. Click on **Next**.
10. DS-5 has a dependency on some external packages, CDT and RSE. Access to the Eclipse website is required to locate and install these packages unless you have already installed them. Click on **Next**.
11. Read the license agreements and accept them. If you do not accept the license agreements, you cannot install DS-5. Click on **Finish**.
12. Click on **Restart Now** to restart Eclipse and complete the installation.
13. If the license is not configured, then a dialog box opens to enable you to add a license. Alternatively, you can configure this later by selecting **Help → ARM License Manager...** from the main menu.

14. If the path to the DS-5 installation is not configured, then a dialog box opens to enable you to locate this folder. Alternatively, you can configure this later by changing the DS-5 settings in the **Window** → **Preferences** dialog box.
15. If you have a full DS-5 installation then you must configure the target database for DS-5:
 - a. Select **Window** → **Preferences** from the main menu.
 - b. Expand the **DS-5** group and select **Target Database**.
 - c. Click on **New...**
 - d. Search the DS-5 installation and select the folder containing the database:
install_directory\DS-5\sw\debugger\configdb.
 - e. Ensure that the new target database is at the top of the list.
 - f. Click on **Rebuild database...** to clear any system caches and validate the platform definitions.
 - g. Click **OK**.

3.1.3 See also

Tasks

- [Managing DS-5 licenses on page 3-36](#)
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Using the welcome screen on page 2-5.](#)

Concepts

- [About DS-5 on page 2-2.](#)

Reference

- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Preferences dialog box on page 2-30.](#)

Other information

- *Java SE Downloads*, <http://www.oracle.com>
- *Eclipse*, <http://www.eclipse.org>.

3.2 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 `ecclipse` 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.

3.2.1 See also

Tasks

- [Building the Gnometris project from Eclipse on page 3-5](#)
- [Building the Gnometris project from the command-line on page 3-6](#)
- [Loading the Gnometris application on a Real-Time System Model on page 3-7](#)
- [Loading the Gnometris application on to an ARM Linux target on page 3-8](#)
- [Debugging Gnometris on page 3-18](#)
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Launching Eclipse on page 2-4](#)
 - [Creating a working set on page 2-17](#)
 - [Changing the top level element when displaying working sets on page 2-20](#)
 - [Deselecting a working set on page 2-21](#)
 - [Using the import wizard on page 2-35.](#)

Concepts

- [About Eclipse for DS-5 on page 2-3](#)
- [Examples on page 4-8](#)
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [About working sets on page 2-16.](#)

Reference

- [Examples on page 4-8.](#)

3.3 Building the Gnetris project from Eclipse

Gnetris 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 Gnetris 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 `gnetris` and `distribution example` projects from the relevant ZIP archive files into Eclipse.
3. Select the `gnetris` project in the Project Explorer view.
4. Select **Build Project** from the **Project** menu.

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

When you build the Gnetris 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 3-4
- [Building the Gnetris project from the command-line](#) on page 3-6
- [Loading the Gnetris application on a Real-Time System Model](#) on page 3-7
- [Loading the Gnetris application on to an ARM Linux target](#) on page 3-8
- [Debugging Gnetris](#) on page 3-18
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Chapter 3 Working with projects.](#)

Reference

- [Examples](#) on page 4-8.

3.4 Building the Gnetris project from the command-line

Gnetris 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 Gnetris 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 gnetris 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\gnetris`.
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 Gnetris 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 Gnetris project from Eclipse on page 3-5](#)
- [Loading the Gnetris application on a Real-Time System Model on page 3-7](#)
- [Loading the Gnetris application on to an ARM Linux target on page 3-8](#)
- [Debugging Gnetris on page 3-18.](#)

Reference

- [Examples on page 4-8.](#)

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

You can load the Gnetris 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 Gnetris:

1. Launch Eclipse.
2. Click on the Project Explorer view.
3. Expand the gnetris project folder.
4. Right-click on the launch file, gnetris-RTSM-example.launch.
5. In the context menu, select **Debug As**.
6. Select the gnetris-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 3-4](#)
- [Building the Gnetris project from Eclipse on page 3-5](#)
- [Building the Gnetris project from the command-line on page 3-6](#)
- [Debugging Gnetris on page 3-18](#)
- *ARM® DS-5™ Using the Debugger:*
 - [Configuring a connection to an RTSM model on page 3-3.](#)

Reference

- [Documentation on page 4-6](#)
- [Examples on page 4-8](#)
- *ARM® DS-5™ Using the Debugger:*
 - [Debug Configurations - Connection tab on page 11-81](#)
 - [Debug Configurations - Files tab on page 11-85](#)
 - [Debug Configurations - Debugger tab on page 11-90](#)
 - [Debug Configurations - Environment tab on page 11-97.](#)

3.6 Loading the Gnetris application on to an ARM Linux target

You can load the Gnetris 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 RSE connection to work with an ARM Linux target on page 3-9](#)
- [Connecting to the Gnetris application that is already running on a ARM Linux target on page 3-15](#)
- [Debugging Gnetris on page 3-18.](#)

Reference

- [Documentation on page 4-6](#)
- [Examples on page 4-8](#)
- *ARM® DS-5™ Using the Debugger*:
 - [Debug Configurations - Connection tab on page 11-81](#)
 - [Debug Configurations - Files tab on page 11-85](#)
 - [Debug Configurations - Debugger tab on page 11-90](#)
 - [Debug Configurations - Environment tab on page 11-97](#)
 - [Target management terminal for serial and SSH connections on page 11-111.](#)

3.7 Using an RSE connection to work with 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:

1. In the Remote Systems view, click on **Define a connection to remote system** in the Remote Systems view toolbar.
2. In the Select Remote System Type dialog box, expand the **General** group and select **Linux**.

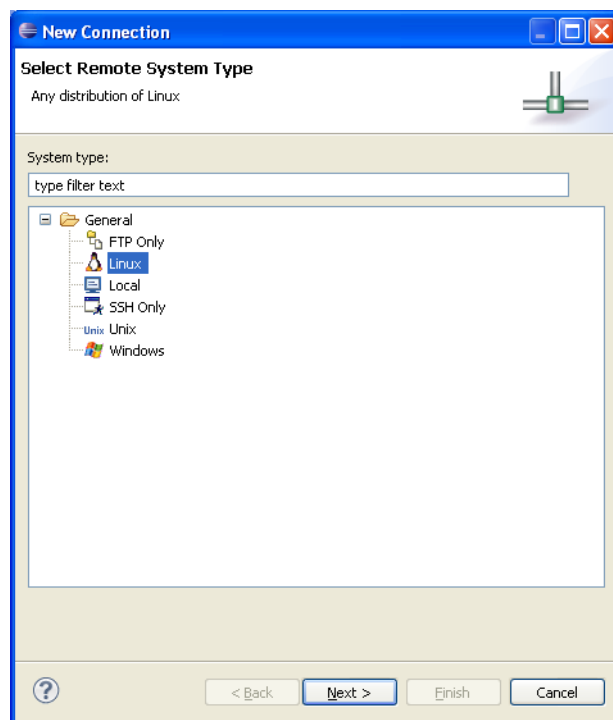


Figure 3-1 Selecting a connection type

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

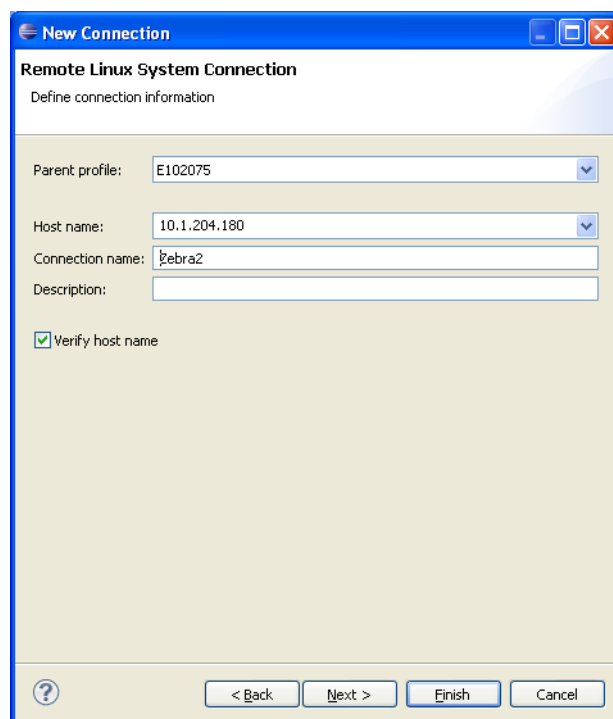


Figure 3-2 Defining the connection information

5. Click **Next**.
6. Select SSH protocol file access.

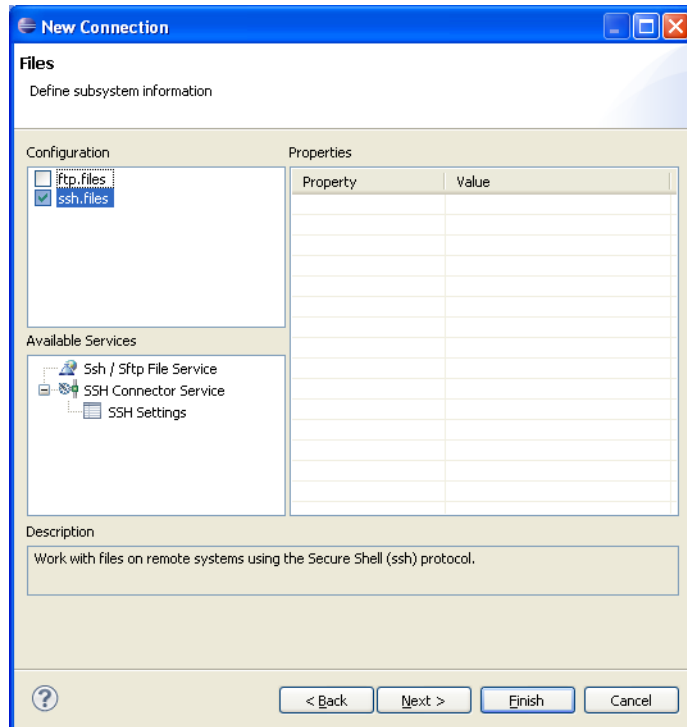


Figure 3-3 Defining the file system

7. Click **Next**.

8. Select the shell processes for Linux systems.

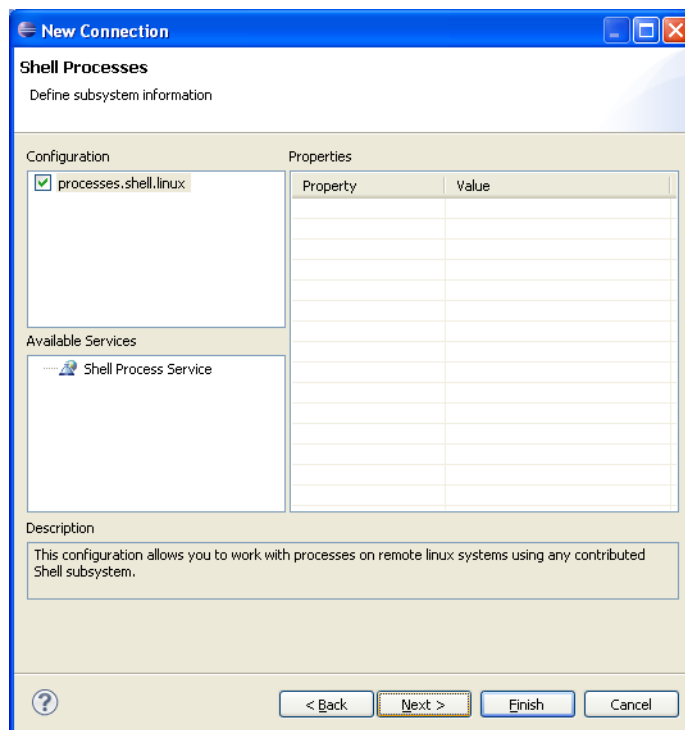


Figure 3-4 Defining the processes

9. Click **Next**.
10. Select SSH shells.

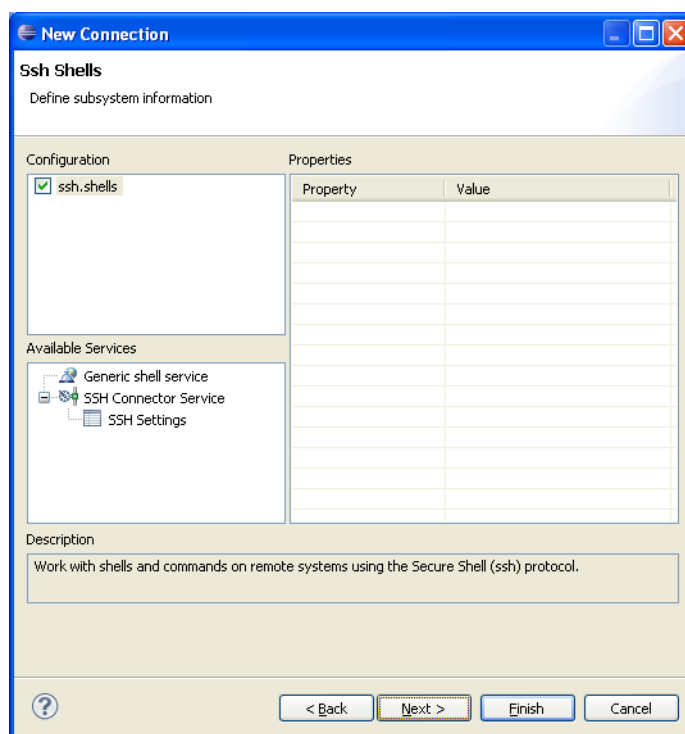


Figure 3-5 Defining the shell services

11. Click **Next**.
12. Select SSH terminals.

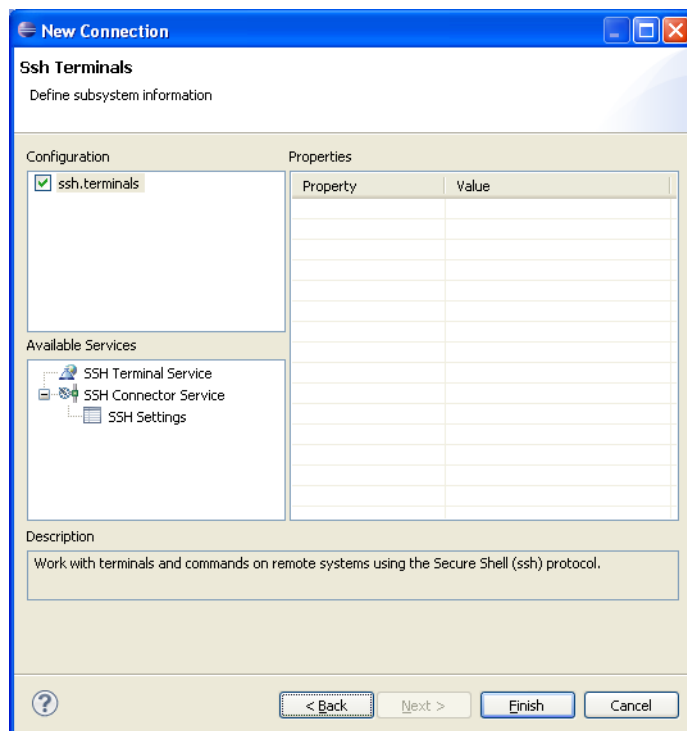


Figure 3-6 Defining the terminal services

13. Click **Finish**.

To connect to the target and run the Gnometriz application:

1. In the Remote Systems view, right-click on the new Linux connection and select **Connect** from the context menu.
2. In the Enter Password dialog box, enter a **User ID** and **Password** if required.
3. Click **OK** to close the dialog box.
4. Copy the stripped version of the Gnometriz application, gnometriz, and the libgames-support.so library from the local file system on to the target file system. You can do this by dragging and dropping the relevant files in the Remote Systems view.
5. Ensure that the files on the target have execute permissions. To do this, right-click on each file in the Remote Systems view, select **Properties** from the context menu and change the checkboxes as required.

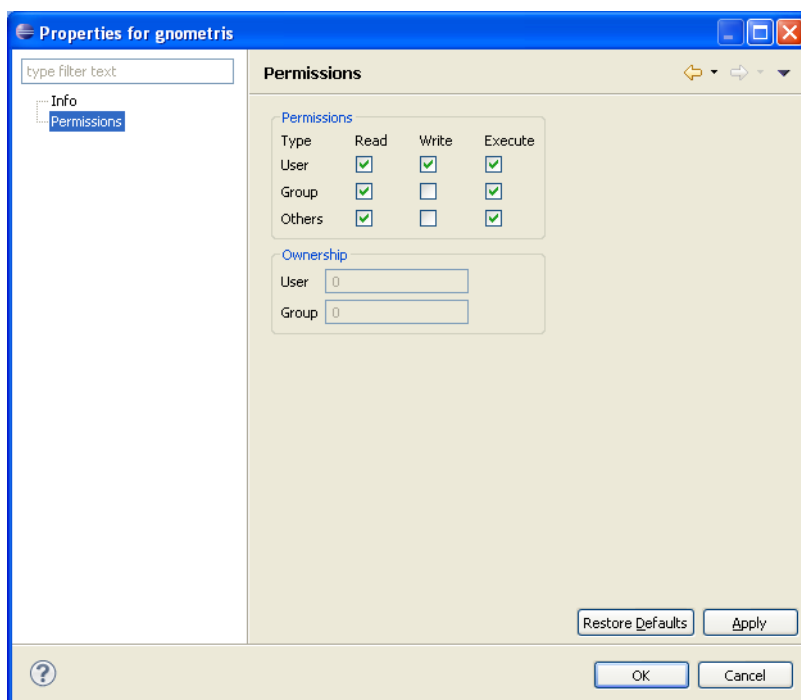


Figure 3-7 Modifying file properties from the Remote Systems view

6. 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 gnometriz application, then execute the following command:


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

 where:

ip is the IP address of the host to display the Gnometriz 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 Gnometriz application that is already running on a ARM Linux target on page 3-15](#)
- [Debugging Gnometriz on page 3-18.](#)

Reference

- [Examples on page 4-8](#)
- *ARM® DS-5™ Using the Debugger:*
 - [Debug Configurations - Connection tab on page 11-81](#)

- *Debug Configurations - Files tab* on page 11-85
- *Debug Configurations - Debugger tab* on page 11-90
- *Target management terminal for serial and SSH connections* on page 11-111.

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

To connect the debugger to the Gnometriz 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.

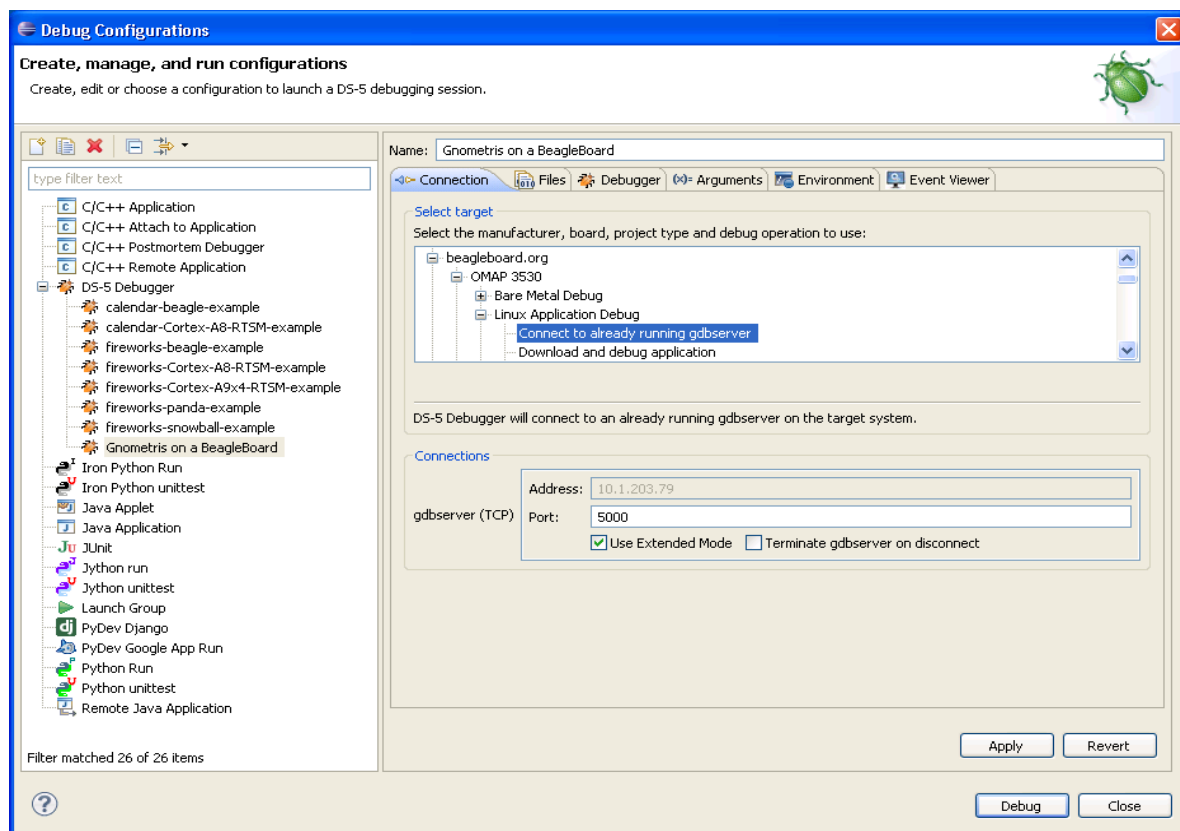


Figure 3-8 Typical connection configuration for a Beagle board

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 Gnometriz application. For example:
H:\workspace\gnometris\libgames-support.so.

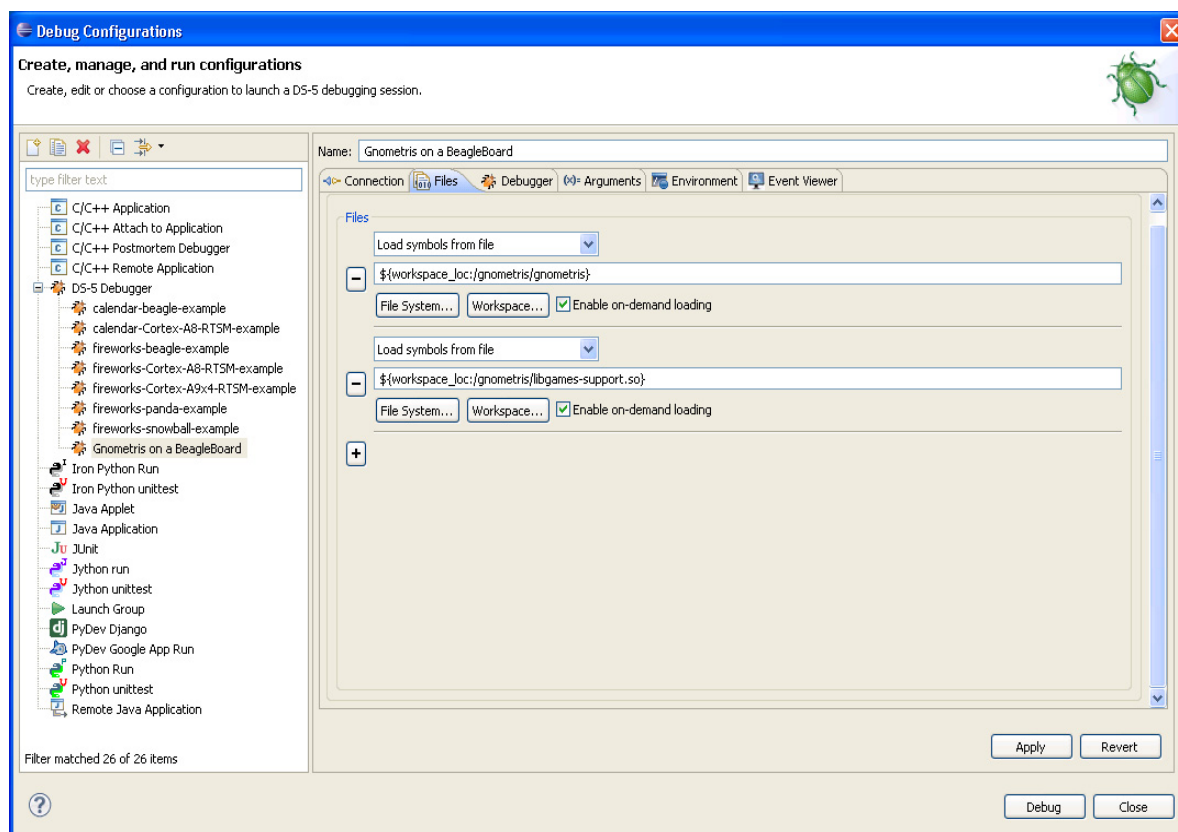


Figure 3-9 Typical file selection for a Beagle board

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**.

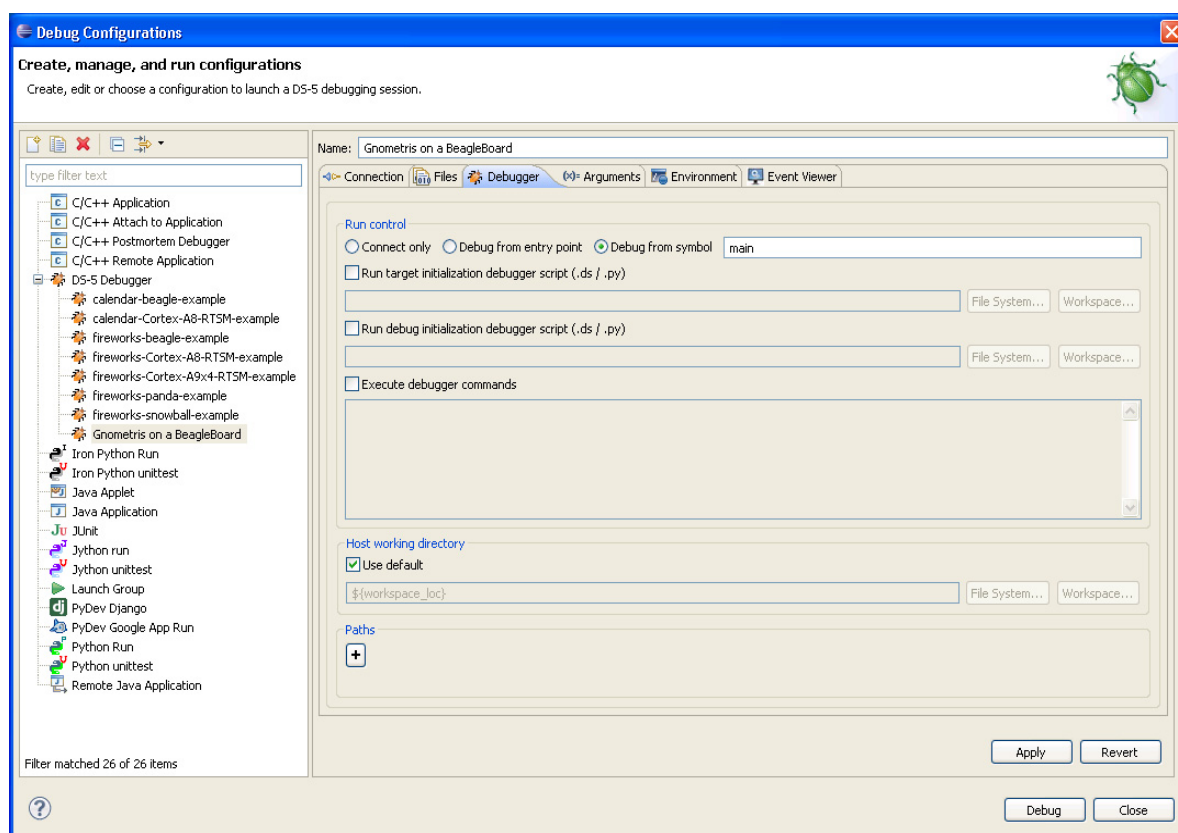


Figure 3-10 Typical debugger settings for a Beagle board

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 3-18.

Reference

- [Examples](#) on page 4-8
- *ARM® DS-5™ Using the Debugger:*
 - [Debug Configurations - Connection tab](#) on page 11-81
 - [Debug Configurations - Files tab](#) on page 11-85
 - [Debug Configurations - Debugger tab](#) on page 11-90.

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-noclobber.cpp` to open the file.
4. In the `blockops-noclobber.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 3-4](#)
- [Building the Gnometris project from Eclipse on page 3-5](#)
- [Building the Gnometris project from the command-line on page 3-6](#)
- [Loading the Gnometris application on a Real-Time System Model on page 3-7.](#)
- [Loading the Gnometris application on to an ARM Linux target on page 3-8.](#)
- *ARM® DS-5™ Using the Debugger:*
 - [Configuring a connection to a Linux target using gdbserver on page 3-5.](#)
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Remote Systems view on page 11-107.](#)

Reference

- [Examples on page 4-8](#)
- *ARM® DS-5™ Using the Debugger:*
 - [C/C++ editor on page 11-13](#)
 - [Debug Control view on page 11-19](#)
 - [Registers view on page 11-44.](#)

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.

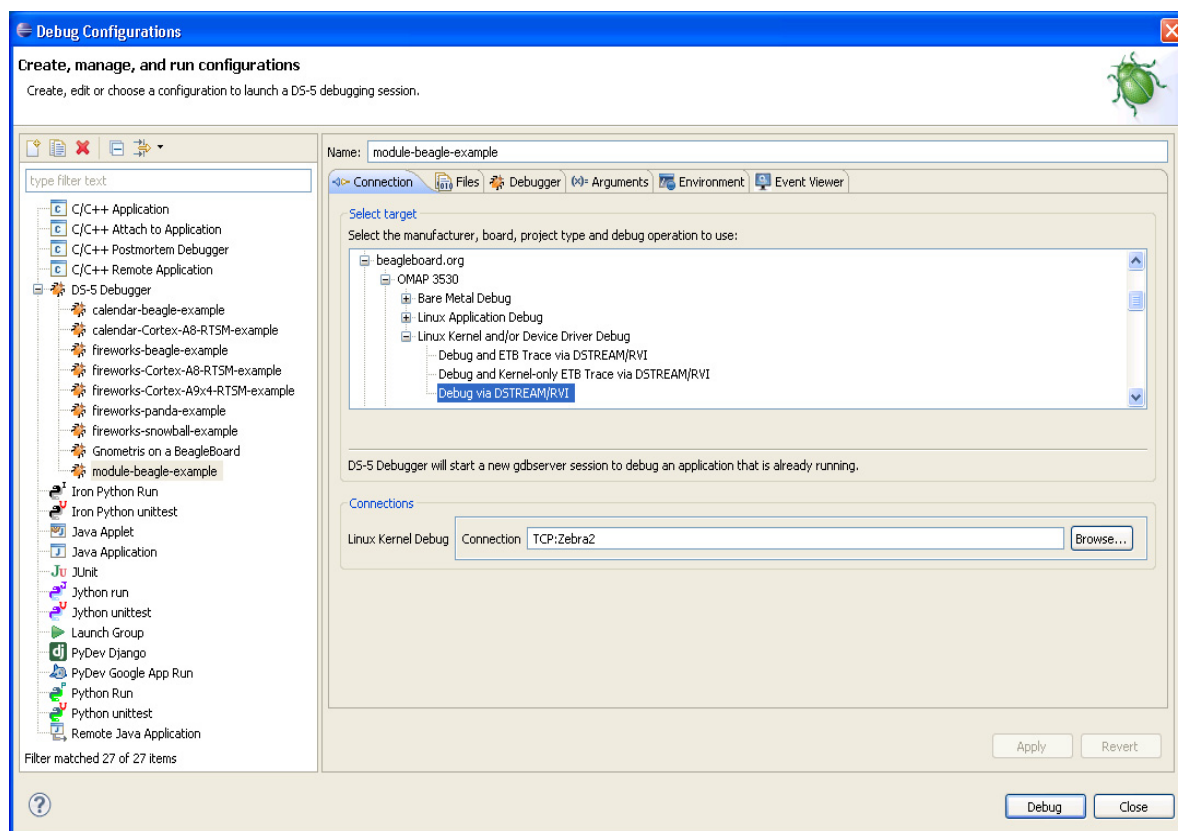


Figure 3-11 Typical connection for a Linux kernel module configuration

- 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.

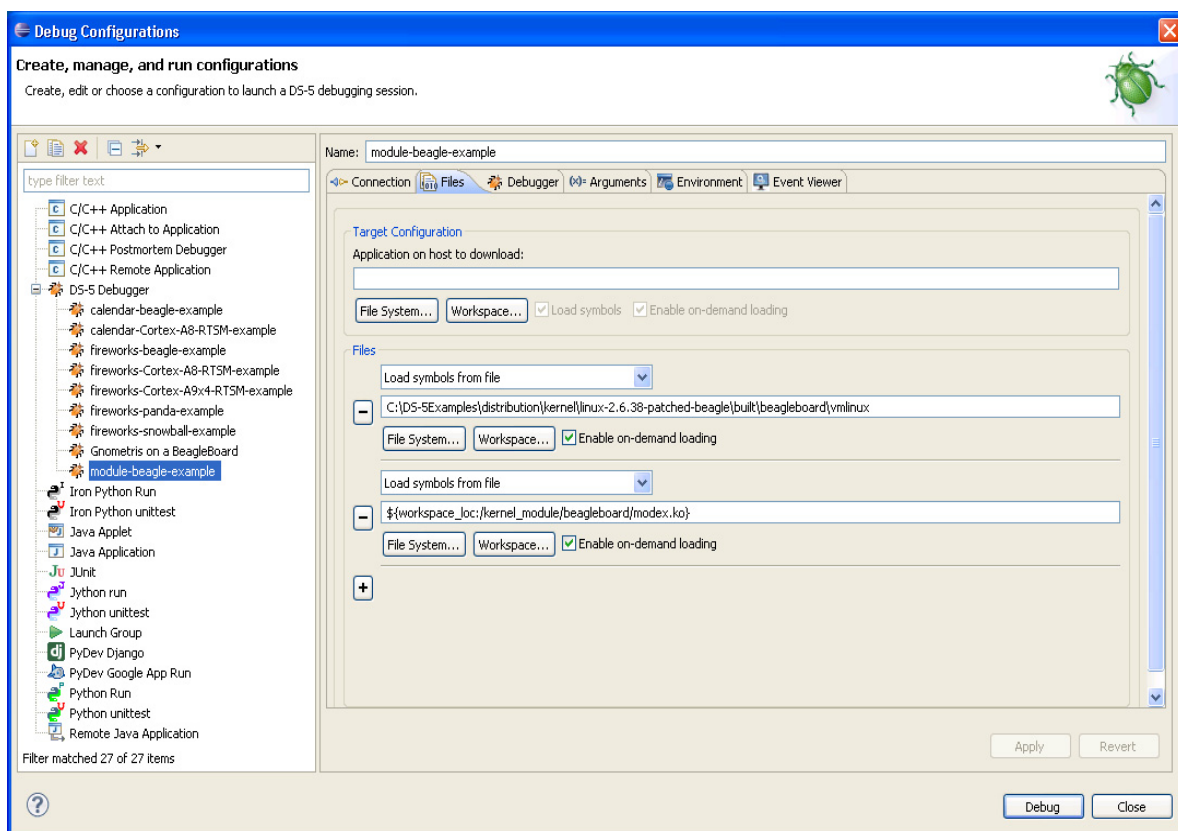


Figure 3-12 Typical file selection for a Linux kernel module configuration

- 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:

<code>lsmod</code>	Displays information about all the loaded modules.
<code>insmod</code>	Inserts a loadable module.
<code>rmmod</code>	Removes a module.

Useful DS-5 Debugger commands:

<code>info os-modules</code>	Displays a list of OS modules loaded after connection.
<code>info os-log</code>	Displays the contents of the OS log buffer.
<code>info os-version</code>	Displays the version of the OS.
<code>info processes</code>	Displays a list of processes showing ID, current state and related stack frame information.
<code>set os-log-capture</code>	Controls the capturing and printing of <i>Operating System</i> (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 5 Controlling execution](#)
 - [Chapter 6 Examining the target.](#)

Concepts

- *ARM® DS-5™ Using the Debugger:*
 - [About debugging a Linux kernel on page 7-15](#)
 - [About debugging Linux kernel modules on page 7-17](#)
 - [ARM Linux problems and solutions on page 12-2](#)
 - [Target connection problems and solutions on page 12-4.](#)

Reference

- [Examples on page 4-8](#)
- *ARM® DS-5™ Using the Debugger:*
 - [Breakpoints view on page 11-9](#)
 - [Commands view on page 11-16](#)
 - [Debug Control view on page 11-19](#)
 - [Modules view on page 11-41.](#)
- *ARM® DS-5™ Debugger Command Reference:*
 - [info os-log on page 2-95](#)

- [*info os-modules*](#) on page 2-96
- [*info os-version*](#) on page 2-97
- [*info processes*](#) on page 2-98
- [*set os*](#) on page 2-158
- [*show os*](#) on page 2-186.
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [*Remote Systems terminal for SSH connections*](#) on page 11-110.

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 4-6](#)
- [Examples on page 4-8](#)
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Remote Systems terminal for SSH connections on page 11-110](#)
- *ARM® DS-5™ Using ARM Streamline,*
<http://infocenter.arm.com/help/topic/com.arm.doc.dui0482-/index.html>.

3.12 Setting up the Android tools for use with DS-5

This tutorial describes the steps required for Android development using DS-5, the Android *Native Development Kit* (NDK), Eclipse with *C/C++ Development Tooling* (CDT), and the Android *Software Development Kit* (SDK) Platform 2.2.

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

Note

Use of the Android development kits are subject to their own terms and conditions.

3.12.1 Procedure

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

1. Download and install the Android SDK tools and required platforms. This enables you to build Java applications together with any native C/C++ code into an Android package with a .apk file extension.

Note

This example uses the Android SDK Platform 2.2.

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 Android NDK ... \toolchains\... \prebuilt directory with the required version of gdbserver provided by DS-5. You can locate this file by selecting **Help** → **ARM Extras...** from the main menu. Ensure that you rename it to gdbserver.
4. Add the ... \android-sdk\platform-tools folder to your PATH environment variable. If it is already configured then you can skip this step.
5. Set up the Eclipse plug-in for Android:
 - a. Launch Eclipse.
 - b. Install the DS-5 Eclipse plug-ins. If you have a full DS-5 installation then you can skip this step.
 - c. Install the *Android Development Tools* (ADT) Eclipse plug-ins.
 - d. Select **Window** → **Preferences** → **Android** and click on **Browse...** to set the location of the Android SDK.
 - e. Open the Android SDK and AVD Manager dialog box by selecting **Window** → **Android SDK and AVD Manager**.
 - f. Expand the **Available packages** group and add SDK platforms as required. For example, Android SDK Platform Android 2.2.

3.12.2 See also

Tasks

- *Installing DS-5 into a custom Eclipse environment* on page 3-2
- *Loading the hello-neon application on to an Android target* on page 3-28
- *Connecting to an application that is already running on an Android target* on page 3-32.

Other information

- *DS-5 Knowledge Articles*,
<http://infocenter.arm.com/help/topic/com.arm.doc.faqs/kiXXwMK1Sxk7vf.html>
- *EclipseFoundation*, <http://eclipse.org>
- *Cygwin*, <http://www.cygwin.com/>
- *Android Developers*, <http://developer.android.com>.

3.13 Loading the hello-neon application on to an Android target

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

Note

Use of the Android development kits are subject to their own terms and conditions.

3.13.1 Procedure

To debug the application:

1. Launch Eclipse.
2. Start the target.
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**.
 - f. Click **Browse...** to locate the hello-neon folder.
 - g. Click **Next**.
 - h. Select the required Build Target. For example, **Android 2.2**.
 - i. 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. Build the hello-neon source files with debug information using the scripts provided by the Android NDK. This tutorial uses the NDK revision 6b. For example:


```
./ndk-build -C samples/hello-neon NDK_TOOLCHAIN=arm-linux-androideabi-4.4.3
NDK_DEBUG=1
```
6. Select the Android project and build the HelloNeon.apk.
7. Create a new *Android Virtual Device* (AVD) and start the emulator.
8. Select **Debug Configurations...** from the **Run** menu.
9. 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.
10. In the Name field, enter a suitable name for the new configuration. For example, HelloNeon.
11. Click on the **Connection** tab to see the target and connection options.

12. In the Select target panel:
 - a. Select **Android - Generic platform**.
 - b. Select **Download and debug an Android application** for the debug operation.
 - c. Select the emulator in the Connections field.

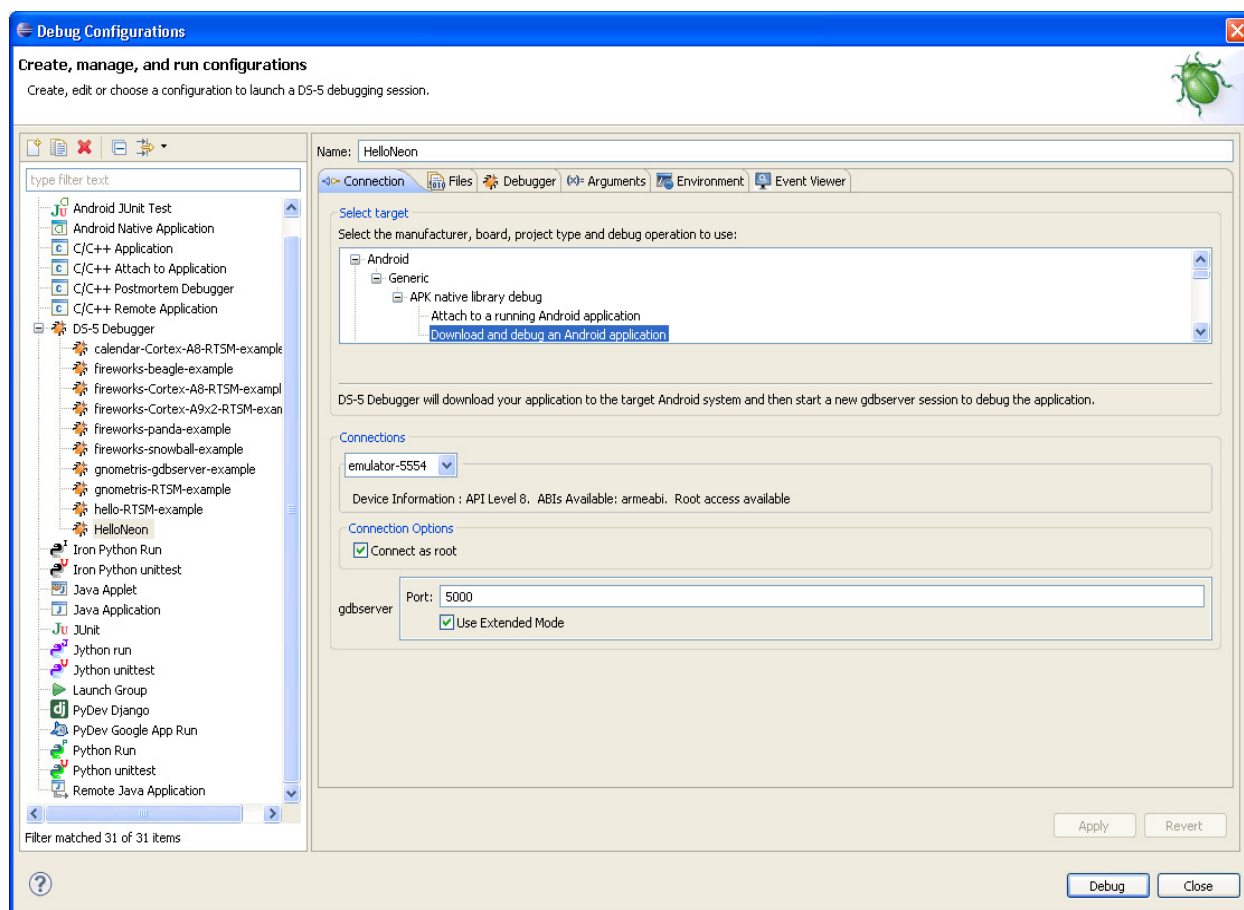


Figure 3-15 Typical Connection tab settings for an Android application

13. In the Files tab, click on **Workspace...** for the Project directory field and select the hello-neon directory. This automatically populates the other fields.

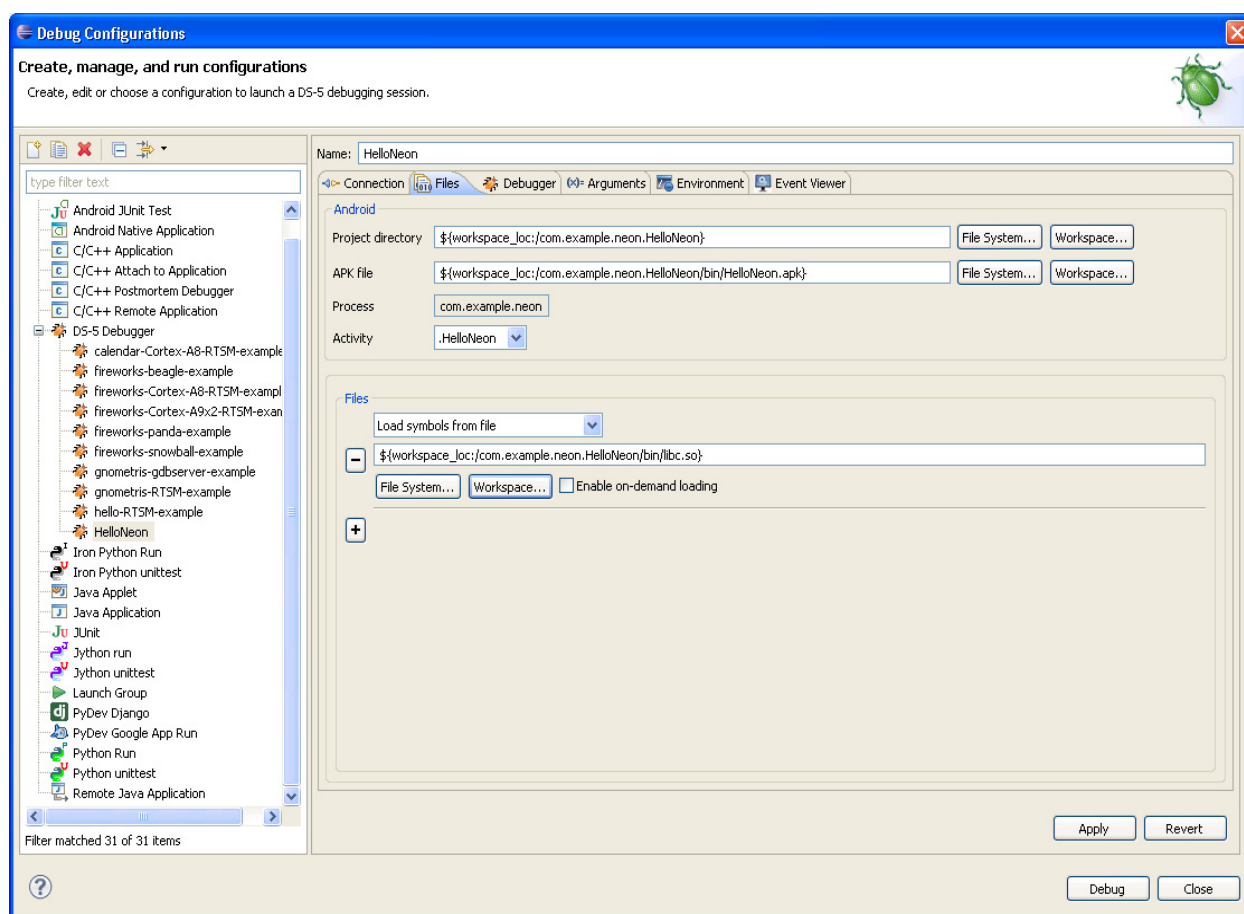


Figure 3-16 Typical Files tab settings for an Android application

14. Click on the **Debugger** tab and select **Connect only** in the Run control panel.
15. In the Paths panel, click on **Workspace...** and select the hello-neon directory.

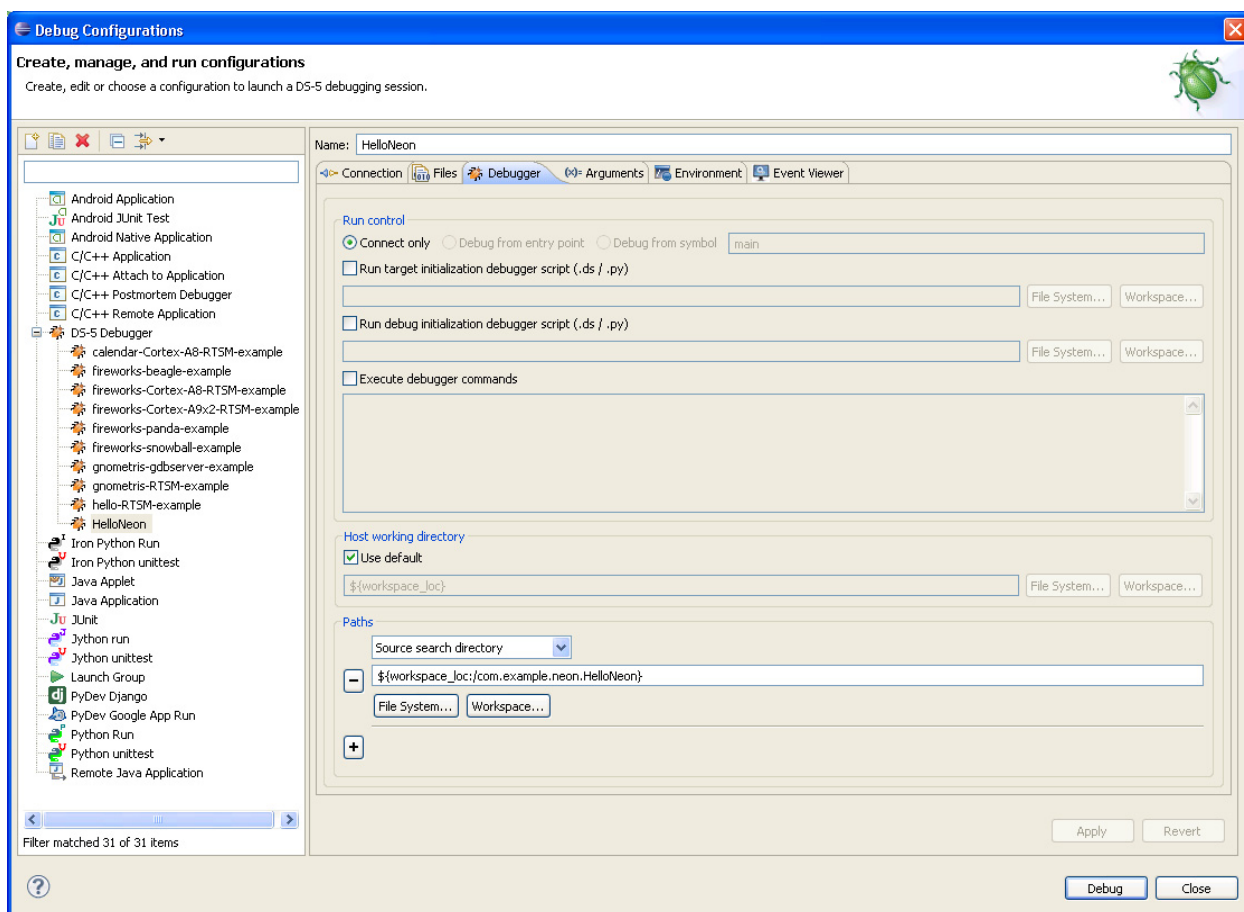


Figure 3-17 Typical Debugger tab settings for an Android application

16. Click on **Debug** to connect to the target.
17. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.
18. To debug the application, set breakpoints, run, and step as required.

———— Note ————

If the application exits before NDK can attach gdbserver to the native library then you might need to add a delay before launching the Java native libraries.

3.13.2 See also

Tasks

- [Setting up the Android tools for use with DS-5 on page 3-26](#)
- [Connecting to an application that is already running on an Android target on page 3-32.](#)

Other information

- *DS-5 Knowledge Articles*, <http://infocenter.arm.com/help/topic/com.arm.doc.faqs/kiXXwMK1Sxk7vf.html>
- *EclipseFoundation*, <http://eclipse.org>
- *Cygwin*, <http://www.cygwin.com/>
- *Android Developers*, <http://developer.android.com>.

3.14 Connecting to an application that is already running on an Android target

This tutorial describes how to connect to and debug an application that is already running on an Android target.

Note

Use of the Android development kits are subject to their own terms and conditions.

3.14.1 Procedure

To debug the application:

1. Launch Eclipse.
2. Connect the host workstation to the target.
3. Ensure that the application is already installed and running on the target.
4. Select **Debug Configurations...** from the **Run** menu.
5. 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.
6. In the Name field, enter a suitable name for the new configuration. For example, HelloNeon.
7. Click on the **Connection** tab to see the target and connection options.
8. In the Select target panel:
 - a. Select **Android - Generic platform**.
 - b. Select **Attach to a running Android application** for the debug operation.
 - c. Select the target in the Connections field.

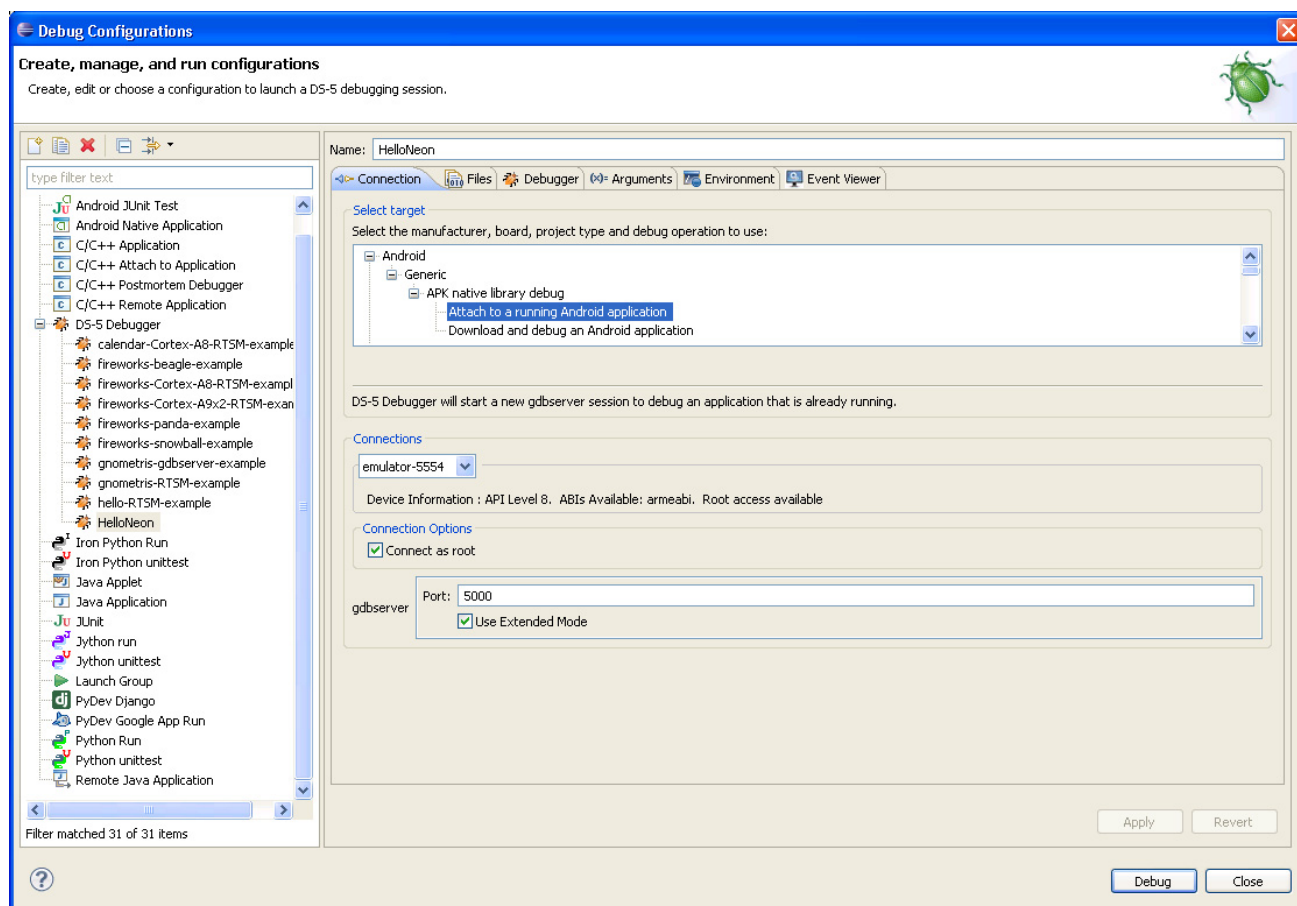


Figure 3-18 Typical Connection tab settings for an Android application

9. In the Files tab, click on **Workspace...** for the Project directory field and select the hello-neon directory. This automatically populates the other fields.

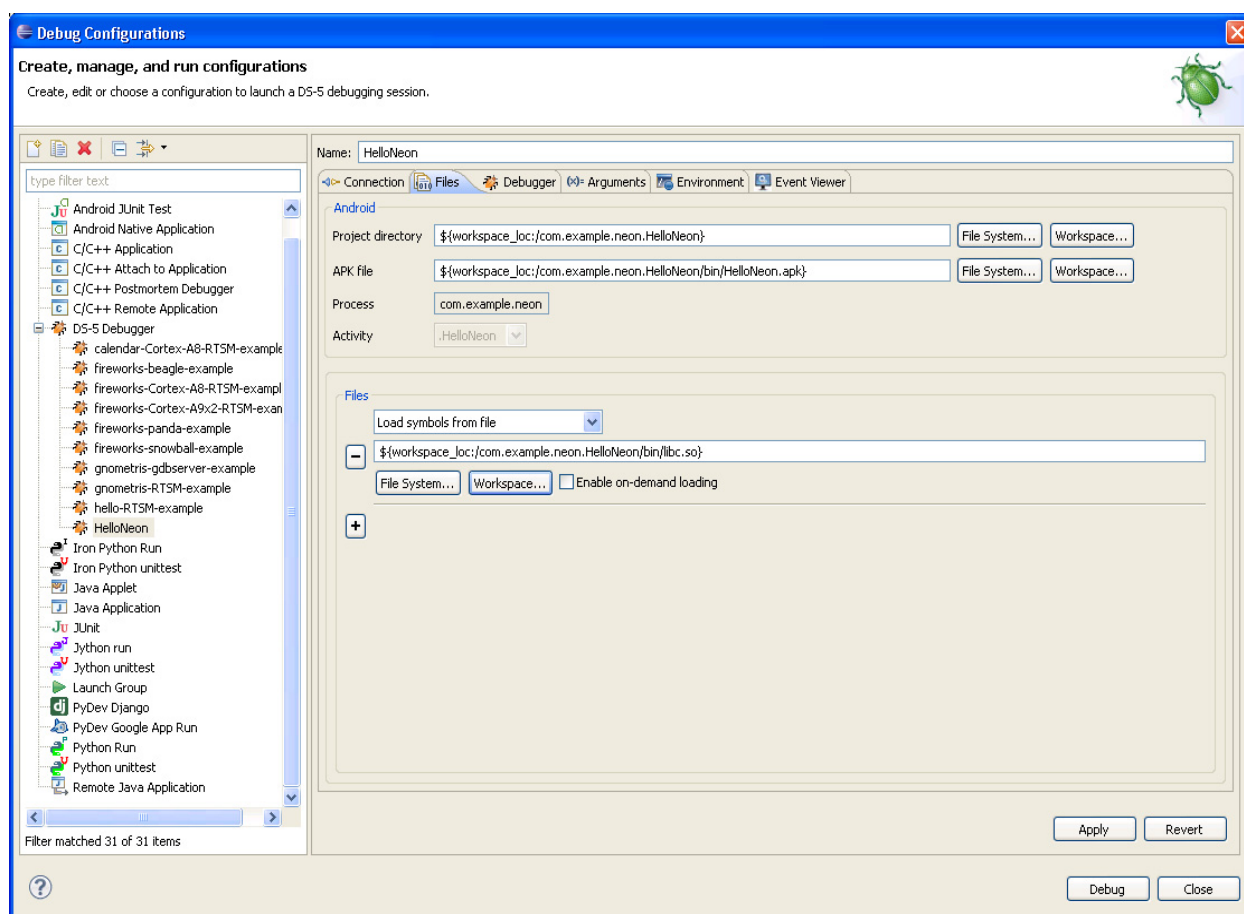


Figure 3-19 Typical Files tab settings for an Android application

10. Click on the **Debugger** tab and select **Connect only** in the Run control panel.
11. In the Paths panel, click on **Workspace...** and select the hello-neon directory.

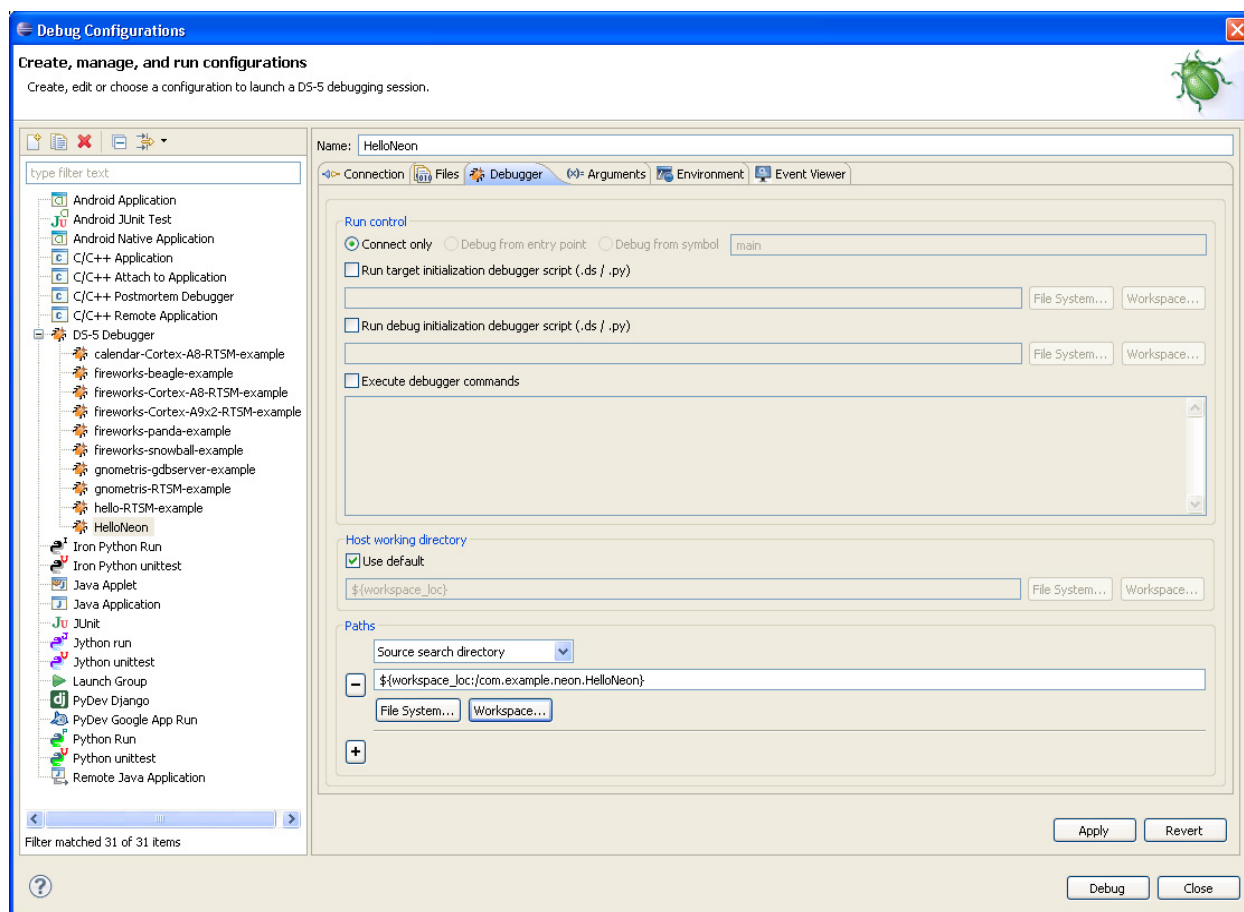


Figure 3-20 Typical Debugger tab settings for an Android application

12. Click on **Debug** to connect to the target.
13. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.
14. To debug the application, set breakpoints, run, and step as required.

———— Note ————

If the application exits before NDK can attach gdbserver to the native library then you might need to add a delay before launching the Java native libraries.

3.14.2 See also

Tasks

- [Setting up the Android tools for use with DS-5 on page 3-26](#)
- [Loading the hello-neon application on to an Android target on page 3-28.](#)

Other information

- *DS-5 Knowledge Articles*, <http://infocenter.arm.com/help/topic/com.arm.doc.faqs/kiXXwMK1Sxk7vf.html>
- *EclipseFoundation*, <http://eclipse.org>
- *Cygwin*, <http://www.cygwin.com/>
- *Android Developers*, <http://developer.android.com>.

3.15 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.

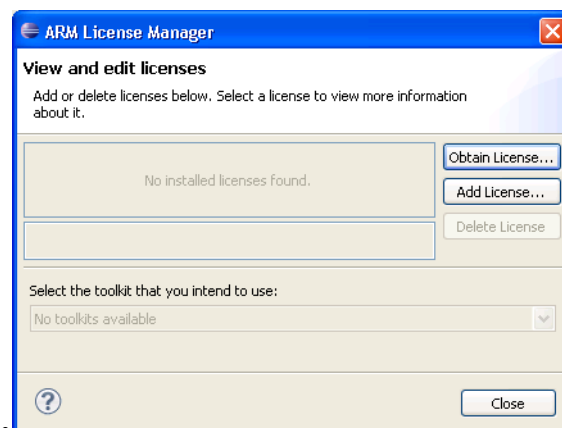


Figure 3-21 View and edit licenses

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

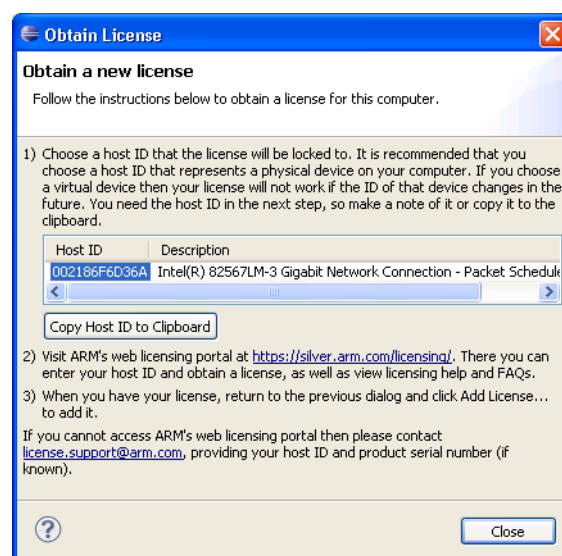


Figure 3-22 Obtain a new license

- 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 3-23 Add a new license

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

3.15.1 See also

Reference

- [Licensing and product updates on page 4-5](#)
- *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/>.

3.16 Changing the Toolkit

To select a toolkit for DS-5:

1. Start Eclipse for DS-5.
2. Select **Help** → **ARM License Manager...** from the Eclipse for DS-5 menu to display the **ARM License Manager** dialog box.
3. Add a license.
4. Select the toolkit from the **Toolkit** drop-down menu to change the type of license that the toolkit checks out.
5. Click **Close**
6. Restart Eclipse for DS-5.

3.16.1 See also

Task

- [Managing DS-5 licenses on page 3-36](#)
- *ARM® DS-5™ Using the Debugger:*
 - [Configuring a connection to an RTSM model on page 3-3](#)
 - [Configuring a connection to a Linux target using gdbserver on page 3-5](#)
 - [Configuring a connection to a Linux Kernel on page 3-7](#)
 - [Configuring a connection to a bare-metal target on page 3-9](#)
 - [Configuring an Event Viewer connection to a bare-metal target on page 3-11.](#)

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

- *System requirements* on page 4-2
- *Installation directories* on page 4-4
- *Licensing and product updates* on page 4-5
- *Documentation* on page 4-6
- *Examples* on page 4-8.

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 service pack 1
- Windows 7 Enterprise service pack 1
- Windows Server 2003 (ARM Compiler only)
- Windows Server 2008 R2 (ARM Compiler only)
- Ubuntu Desktop Edition 12.04 LTS
- 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 7.0 or later. An executable for ARM Linux and Android that are compatible with DS-5 Debugger is provided with DS-5. You can locate these files by selecting **Help** → **ARM Extras...** from the main menu.

———— Note ————

DS-5 Debugger is unable to provide reliable multi-threaded debug support with gdbserver versions prior to 6.8.

DS-5 Debugger supports debugging native C/C++ applications and libraries on Android versions 2.2.x, 2.3.x, 3.x.x, and 4.0.

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. Access to VFP and NEON registers requires ARM Linux kernel version 2.6.30 or later and gdbserver version 7.0 or later.

ARM Linux kernel and bare-metal debugging require the use additional tools (not supplied with DS-5) to connect to your target system. DSTREAM and RVI enable connection to physical hardware targets. VSTREAM enables connection to RTL simulators and hardware emulators. For DSTREAM and RVI it is recommended that you use the supplied debug hardware update tool to check that you are using the latest firmware and update it if necessary. Updated firmware is available in the *install_directory/sw/debughw/firmware* directory. The firmware for VSTREAM is delivered as part of the VSTREAM software, so you must obtain and install a newer version of this product if the firmware version is too old.

DS-5 can co-exist with the ARM RVDS toolchain provided that they are installed into separate directories.

All line drawings in the online help use SVG format. To view these graphics, your browser must support the SVG format. If your browser does not have native support for SVG, you must install an appropriate plug-in such as the Adobe SVG Viewer.

4.1.3 See also

Reference

- [About Debug hardware configuration utilities on page 2-9](#)
- [Installation directories on page 4-4](#)
- [Licensing and product updates on page 4-5](#)
- *ARM® DS-5™ Setting up the ARM® DSTREAM™ Hardware*,
<http://infocenter.arm.com/help/topic/com.arm.doc.duit0481-/index.html>
- *ARM® DS-5™ Setting up the ARM® RVT™ and RVT™ Hardware*,
<http://infocenter.arm.com/help/topic/duit0515-/index.html>

Other information

- *DS-5 Knowledge Articles*,
<http://infocenter.arm.com/help/topic/com.arm.doc.faqs/kiXXwMK1Sxk7vf.html>
- *Adobe SVG Viewer*, <http://www.adobe.com/svg/viewer/install>.

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
<i>install_directory</i>	C:\Program Files\DS-5	~/ds-5
<i>arm_directory</i>	<i>install_directory</i> \arm\...	<i>install_directory</i> /arm/...
<i>examples_directory</i>	<i>install_directory</i> \examples\...	<i>install_directory</i> /examples/...
<i>tools_directory</i>	<i>install_directory</i> \bin\...	<i>install_directory</i> /bin/...
<i>documents_directory</i>	<i>install_directory</i> \documents\...	<i>install_directory</i> /documents/...

4.2.1 See also

Reference

- [Documentation on page 4-6](#)
- [Examples on page 4-8.](#)

4.3 Licensing and product updates

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

Table 4-2 DS-5 Editions

	Basic Edition	Professional Edition
Eclipse for DS-5	X	X
ARM Streamline Performance Analyzer	X	X
Android native library debug	X	X
GNU Compilation Tools	X	X
Linux application debug	X	X
Cortex™-A8 real-time system model	X	X
Kernel space debug and trace	X	X
Bare-metal debug and trace	X	X
Cortex-A9 real-time system model		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 3-36.](#)

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/>.
- *ARM forums*, <http://forums.arm.com/>.

4.4 Documentation

The ARM® DS-5™ documentation suite comprises:

- *Getting Started with DS-5* (this document)
- *Using the Debugger*
- *Debugger Command Reference*
- *Using ARM Streamline*
- *Getting Started with Eclipse*
- Debug hardware:
 - *Setting Up the ARM® DSTREAM™ Hardware*
 - *Setting Up the ARM® RVT™ and RVT™ Hardware*
 - *DSTREAM™ System and Interface Design Reference*
 - *RVT™ and RVT™ System and Interface Design Reference*
 - *Using the Debug Hardware Configuration Utilities*
- *Model Shell for Fast Models Reference Manual*
- *Real-Time System Model Reference*
- ARM Compiler toolchain:
 - *Introducing the ARM Compiler toolchain*
 - *Developing Software for ARM® Processors*
 - *Using the Compiler*
 - *Using the Assembler*
 - *Using the Linker*
 - *Using ARM® C and C++ Libraries and Floating-Point Support*
 - *Creating Static Software Libraries with armar*
 - *Using the fromelf Image Converter*
 - *Building Linux Applications with the ARM® Compiler toolchain and GNU Libraries*
 - *Assembler Reference*
 - *Compiler Reference*
 - *Linker Reference*
 - *ARM® C and C++ Libraries and Floating-Point Support Reference*
 - *Errors and Warnings Reference*
 - *Migration and Compatibility*
- *License Management Guide.*

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 4-4](#)
- [Examples on page 4-8](#)

- *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.
 - Linux distribution project with header files and libraries for the purpose of rebuilding the ARM Linux examples.
 - Linux SD card image for the BeagleBoard configured for DS-5.
 - Linux SD card image for the BeagleBoard-xM configured for DS-5.

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 3-4](#)

Concepts

- [About Real-Time System Models on page 2-5.](#)

Reference

- [Documentation on page 4-6](#)
- [Installation directories on page 4-4](#)
- *ARM® DS-5™ Getting Started with Eclipse:*
 - [Using the welcome screen on page 2-5.](#)

Other information

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