ARM® DS-5™

Version 5.2

Getting Started with DS-5

ARM®

ARM DUI 0478B (ID100410)

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

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 0478B
- 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.

Chapter 2
ARM DS-5 product overview

The following topics give an overview of ARM® DS-5™.

Concepts

2.1 About DS-5

ARM® DS-5™ is a set of professional tools for software development on ARM processor-based targets. The tools enable you to write, build, run and debug applications on a hardware target or a software model that simulates a hardware target.

DS-5 includes:

- Eclipse for DS-5. An *Integrated Development Environment* (IDE) that combines
- the Eclipse IDE from the Eclipse Foundation with compilation and debug tools.
- GCC Compiler toolchain
- DS-5 Debugger for:
  - Linux applications
  - bare metal
  - bare metal with an *Embedded Trace Buffer* (ETB)
- 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.

2.1.1 See also

Concepts

- *Eclipse for DS-5* on page 2-3
- *DS-5 Debugger* on page 2-4
- *GNU compilation tools* on page 2-6
- *ARM Streamline Performance Analyzer* on page 2-7
- *Debug hardware configuration utilities* on page 2-8.

Reference

- *DS-5 documentation* on page 4-5
- *DS-5 examples* on page 4-6.

2.2 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

- *Eclipse for DS-5 Using Eclipse*:
  - Chapter 3 *Getting started with Eclipse*.

Concepts

- *About DS-5* on page 2-2
- *DS-5 Debugger* on page 2-4
- *GNU compilation tools* on page 2-6
- *ARM Streamline Performance Analyzer* on page 2-7
- *Debug hardware configuration utilities* on page 2-8.

2.3 DS-5 Debugger

DS-5™ Debugger is a graphical debugger supporting end-to-end software development on ARM® processor-based targets. 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.

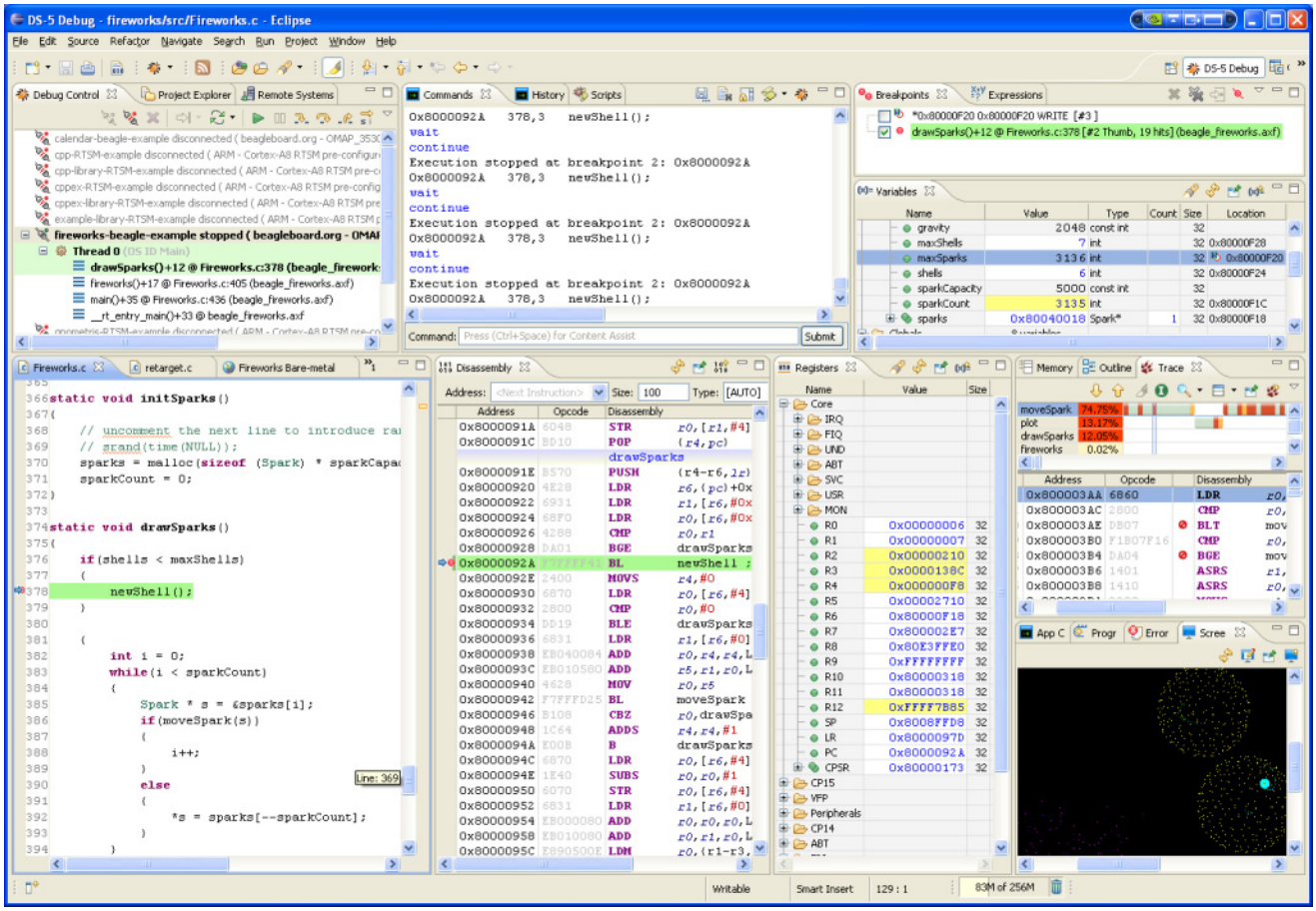

Figure 2-1 DS-5 Debug perspective

Using the Debug Control view you can single step through either at source level or isntruction 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 DS-5* on page 2-2
- *Eclipse for DS-5* on page 2-3

2.4 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 2-1 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.

2.4.1 See also

Tasks

- *Creating a new C or C++ project in Eclipse* on page 3-5
- *Building the Gnometris project from Eclipse* on page 3-6
- *Building the Gnometris project from the command-line* on page 3-7.

Concepts

- *About DS-5* on page 2-2
- *Eclipse for DS-5* on page 2-3
- *DS-5 Debugger* on page 2-4
- *ARM Streamline Performance Analyzer* on page 2-7
- *Debug hardware configuration utilities* on page 2-8.

2.5 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. ARM Streamline uses hardware performance counters with kernel metrics to provide an accurate representation of system resources.

2.5.1 See also

Concepts

- *About DS-5* on page 2-2
- *Eclipse for DS-5* on page 2-3
- *DS-5 Debugger* on page 2-4
- *GNU compilation tools* on page 2-6
- *Debug hardware configuration utilities* on page 2-8.

Reference

- *ARM Streamline Performance Analyzer Quick Start Guide*, http://infocenter.arm.com/help/topic/com.arm.doc.dui0482-/index.html.

2.6 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.

2.6.1 See also

Concepts

- *About DS-5* on page 2-2
- *Eclipse for DS-5* on page 2-3
- *DS-5 Debugger* on page 2-4
- *GNU compilation tools* on page 2-6
- *ARM Streamline Performance Analyzer* on page 2-7.

Reference

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

Chapter 3
DS-5 tutorials

The following tutorials show you how to run and debug applications using 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. It is recommended 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 **File → Import...** to open the Import dialog box.
4. Expand the **General** group.
5. Select **Existing Projects into Workspace**.
6. Click **Next**.
7. Open the examples ZIP archive provided with DS-5:
   a. Click on **Select archive file**.
   b. Click on **Browse...** and locate the archive file, *`examples_directory`*`\examples.zip`.
   c. Click **Open** to open the archive file and close the dialog box. The Projects panel is populated with a list of all the example projects.
   d. Select the required projects. It is strongly recommended that you import all of the projects, to ensure that all dependencies are met.

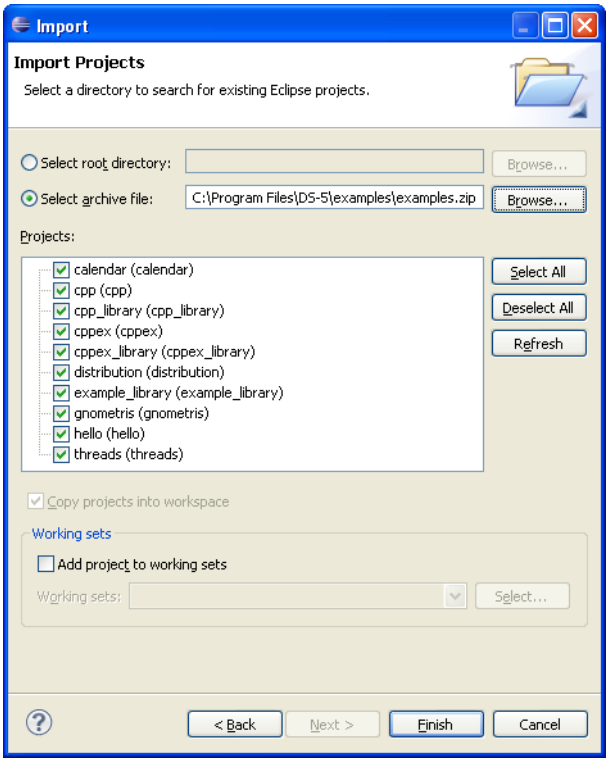

Figure 3-1 Importing the example projects

8. If you are not using working sets to group your projects then you can skip this step.
   a. Select **Add project to working sets**.
   b. Click on **Select...**.
   c. Select an existing working set or create a new one and then select it.
   d. Click **OK**.
9. Click **Finish**.

3.1.1 See also

Tasks

- *Building the Gnometris project from Eclipse* on page 3-6
- *Building the Gnometris project from the command-line* on page 3-7
- *Loading the Gnometris application on a Real-Time System Model* on page 3-8
- *Loading the Gnometris application on to an ARM Linux target* on page 3-9
- *Debugging Gnometris* on page 3-18
- *Debugging the calendar application using Eclipse* on page 3-20
- *ARM DS-5 Using Eclipse*:
  - *Launching Eclipse* on page 3-3
  - *Creating a working set* on page 3-15
  - *Changing the top level element when displaying working sets* on page 3-17
  - *Deselecting a working set* on page 3-18
  - *Using the import wizard* on page 3-33.

Concepts

- *Eclipse for DS-5* on page 2-3

- *DS-5 examples* on page 4-6
- *ARM DS-5 Using Eclipse*:
  - *About working sets* on page 3-14.

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. Click on **Next**.
5. Enter a project name.
6. 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.
7. Select the type of project that you want to create.
8. 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. Import both the `gnometris` and `distribution` example projects into Eclipse.
2. Select the `gnometris` project in the Project Explorer view.
3. 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 3-7
- *Loading the Gnometris application on a Real-Time System Model* on page 3-8
- *Loading the Gnometris application on to an ARM Linux target* on page 3-9
- *Debugging Gnometris* on page 3-18
- *ARM DS-5 Using Eclipse*:
  - Chapter 4 *Working with projects*.

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. Extract both the `gnometris` and `distribution` example projects from the archive file, *`examples_directory`*`\examples.zip` into a working directory.
2. Open the **DS-5 Command Prompt** command-line console.
3. Navigate to `...\ARMLinux\gnometris`.
4. 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 3-6
- *Loading the Gnometris application on a Real-Time System Model* on page 3-8
- *Loading the Gnometris application on to an ARM Linux target* on page 3-9
- *Debugging Gnometris* on page 3-18.

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. For more information on the ARM Linux distribution provided with DS-5 see:
`documents_directory``\examples\ARMLinux\distribution\readme.html`.

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 3-6
- *Building the Gnometris project from the command-line* on page 3-7
- *Debugging Gnometris* on page 3-18
- *ARM DS-5 Using the Debugger*:
  - *Configuring a connection to an RTSM model* on page 3-3.

Reference

- *DS-5 documentation* on page 4-5
- *ARM DS-5 Using the Debugger*:
  - *Debug Configurations - Connection tab* on page 11-55
  - *Debug Configurations - Files tab* on page 11-58
  - *Debug Configurations - Debugger tab* on page 11-62
  - *Debug Configurations - Environment tab* on page 11-67.

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 3-10
- *Connecting to the Gnometris application that is already running on a ARM Linux target* on page 3-15
- *Debugging Gnometris* on page 3-18.

Reference

- *DS-5 documentation* on page 4-5
- *ARM DS-5 Using the Debugger*:
  - *Debug Configurations - Connection tab* on page 11-55
  - *Debug Configurations - Files tab* on page 11-58
  - *Debug Configurations - Debugger tab* on page 11-62
  - *Debug Configurations - Environment tab* on page 11-67.

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**.

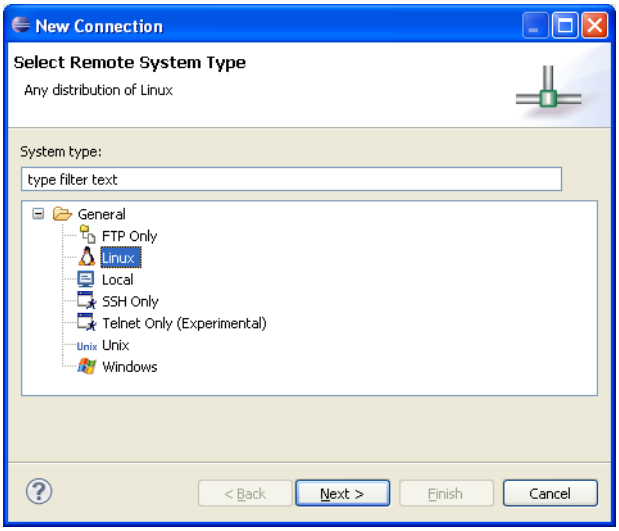

Figure 3-2 Selecting a connection type

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

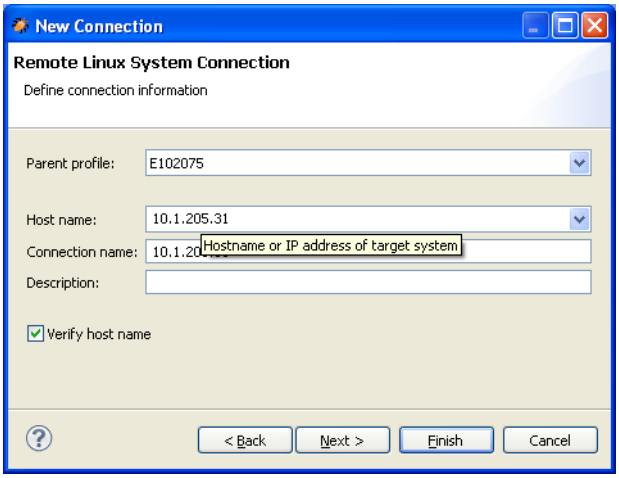

Figure 3-3 Defining the connection information

e. Click **Next**.

f. Select SSH protocol file access.

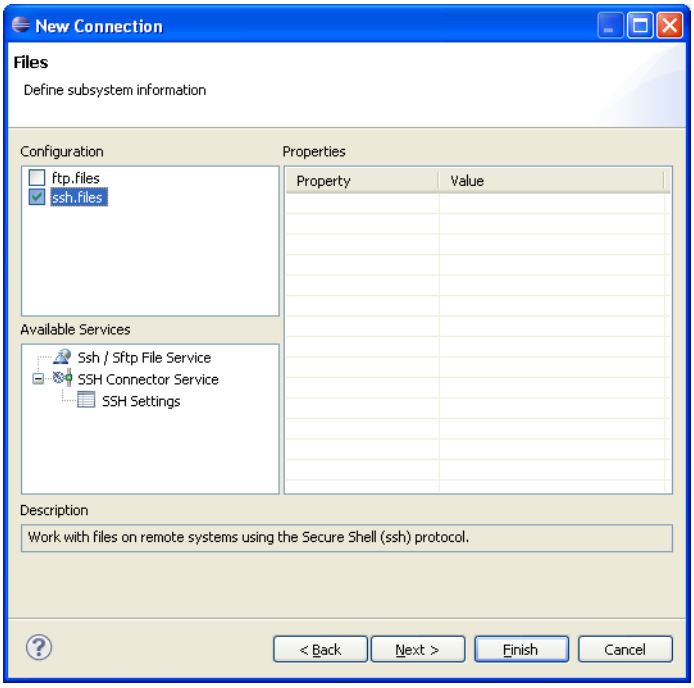

Figure 3-4 Defining the file system

g. Click **Next**.

h. Select the shell processes for Linux systems.

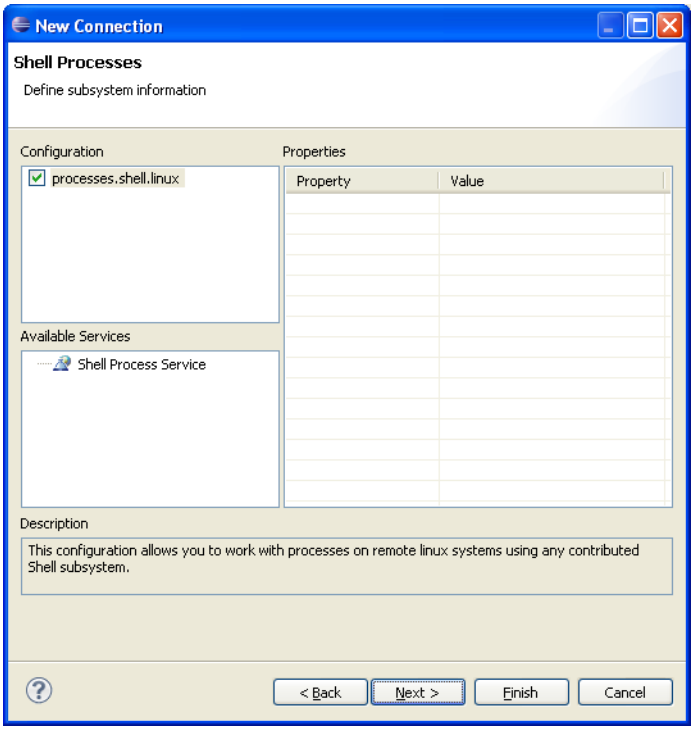

Figure 3-5 Defining the processes

i. Click **Next**.

j. Select SSH shells.

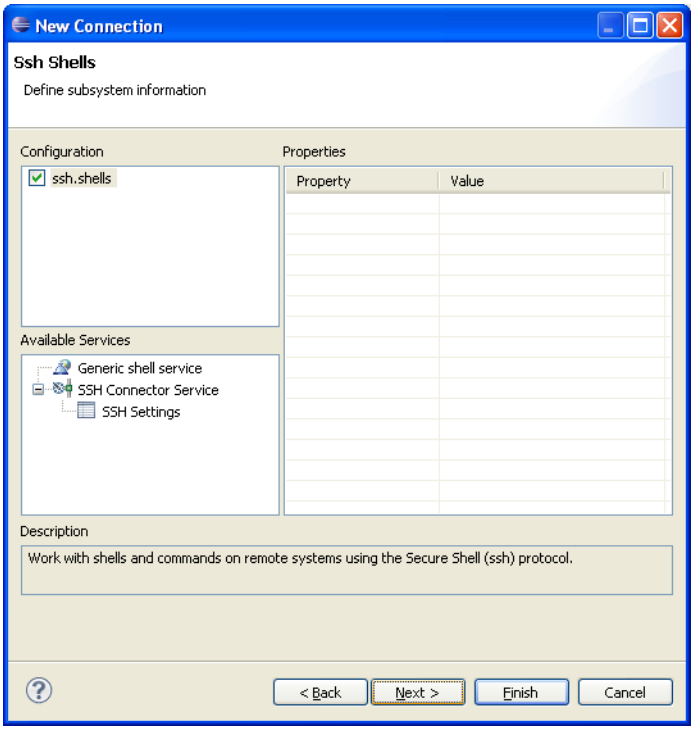

Figure 3-6 Defining the shell services

k. Click **Next**.

l. Select SSH terminals.

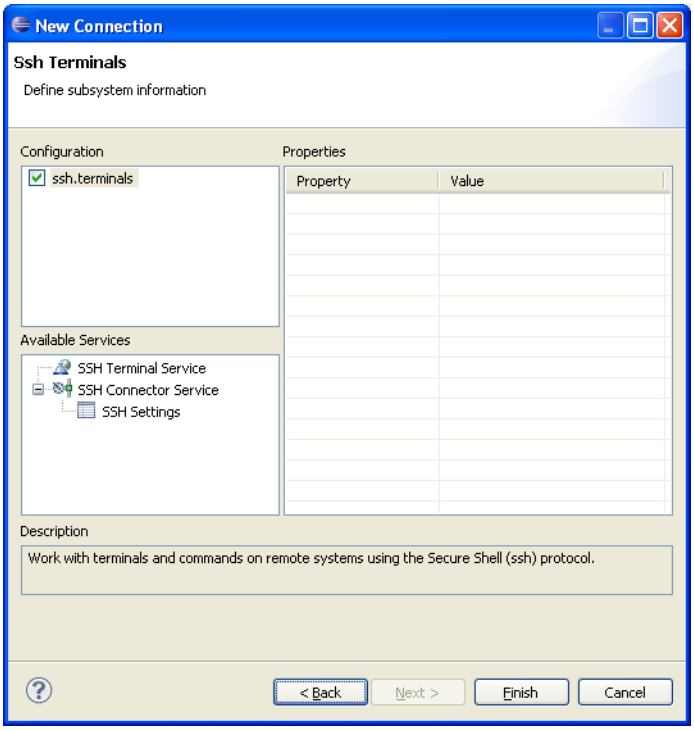

Figure 3-7 Defining the terminal services

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 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.

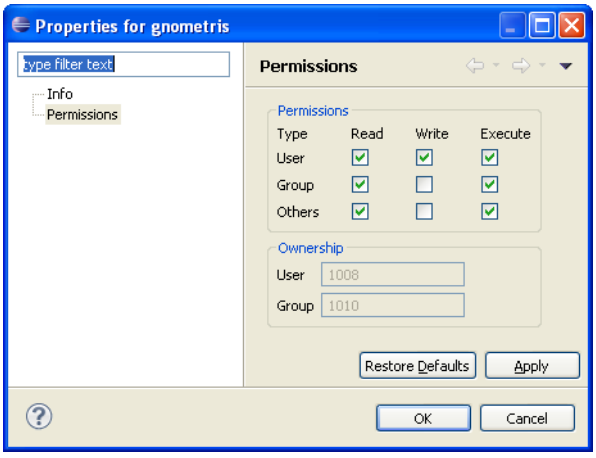

Figure 3-8 Modifying file properties from the Remote Systems view

4. Open a terminal shell that is connected to the target and launch gdbserver with the application:
   a. In the Remote Systems view you can 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 3-15
- *Debugging Gnometris* on page 3-18.

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. Click on the **Connection** tab to see the target and connection options.
3. 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.
4. 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.

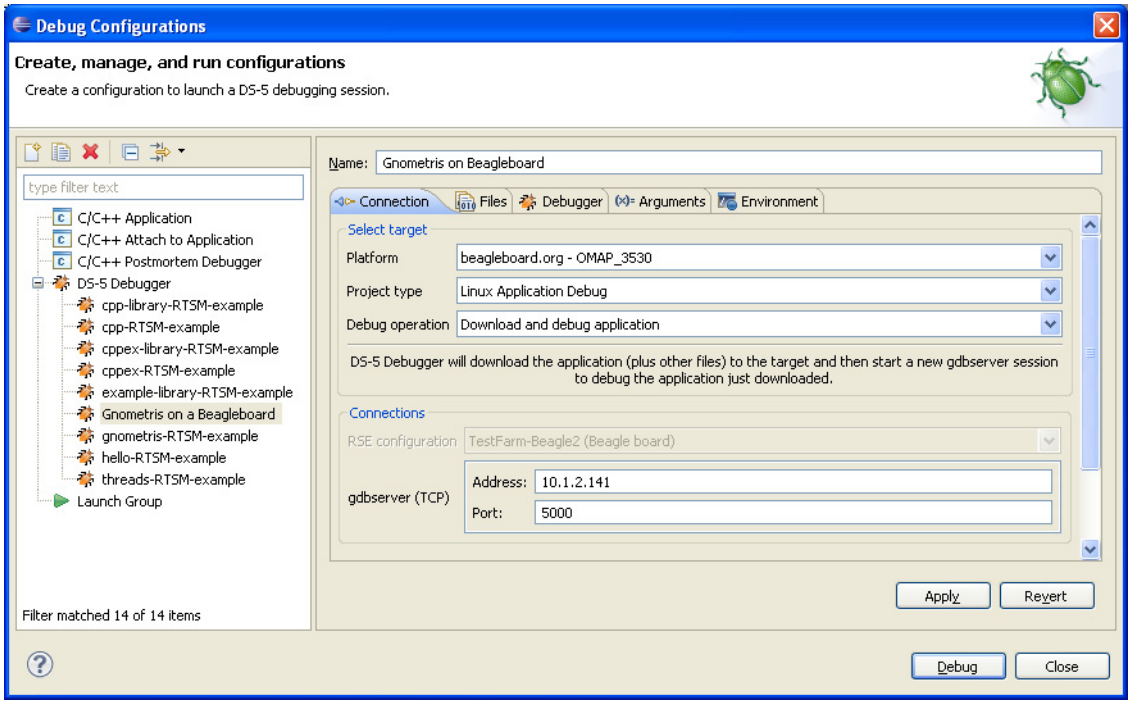

Figure 3-9 Connection tab for a Beagle board

5. Click on the **Files** tab to see the file options.
6. 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`.

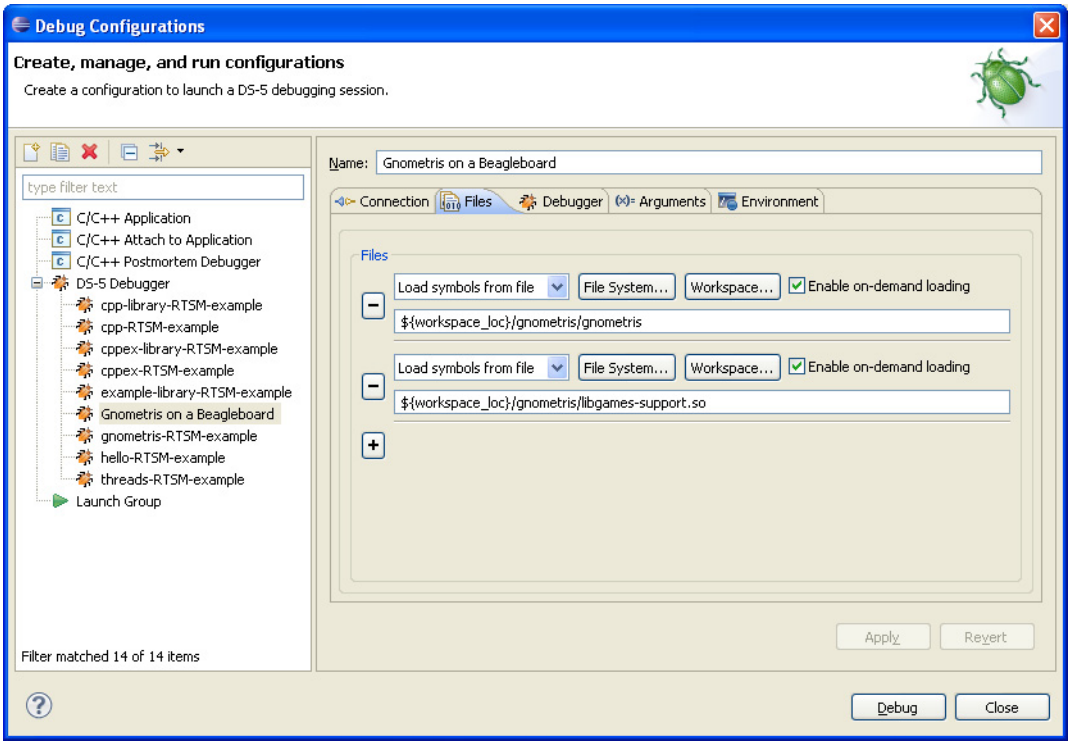

Figure 3-10 Files tab for a Beagle board

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

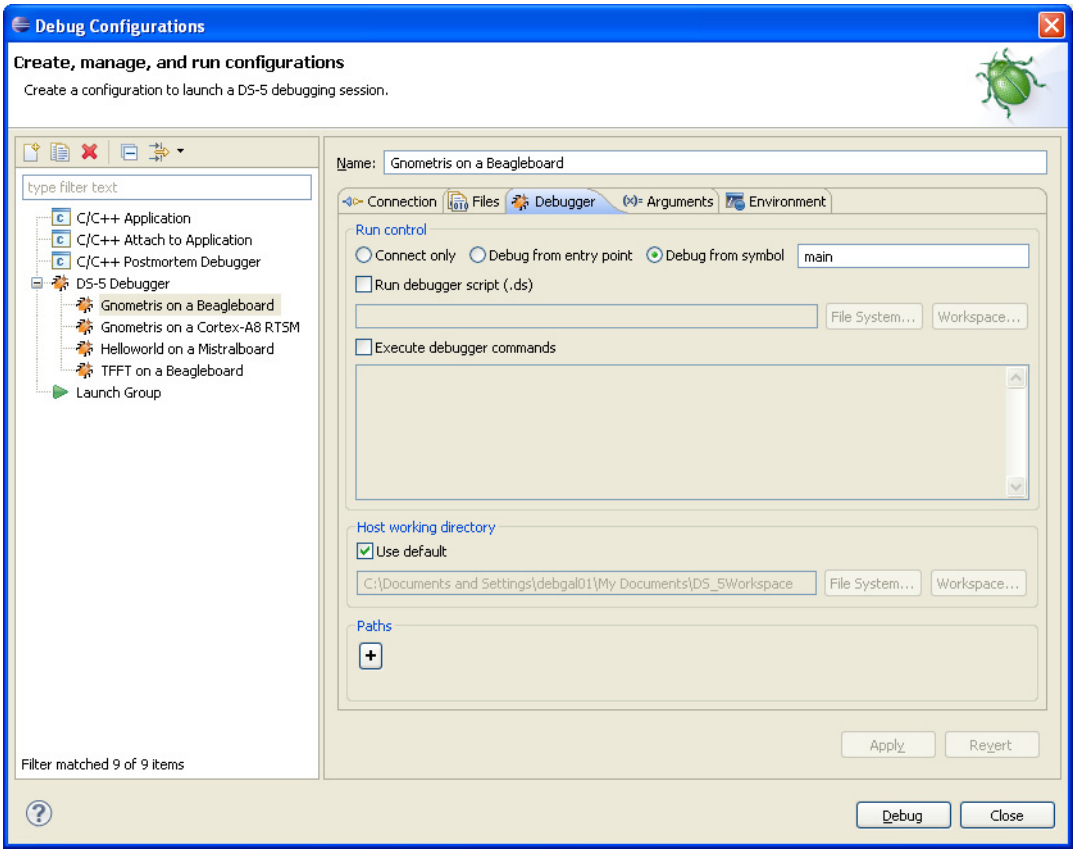

Figure 3-11 Debugger tab for a Beagle board

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

11. 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

- *ARM DS-5 Using the Debugger*:
  - *Debug Configurations - Connection tab* on page 11-55
  - *Debug Configurations - Files tab* on page 11-58
  - *Debug Configurations - Debugger tab* on page 11-62.

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 3-6
- *Building the Gnometris project from the command-line* on page 3-7
- *Loading the Gnometris application on a Real-Time System Model* on page 3-8.
- *Loading the Gnometris application on to an ARM Linux target* on page 3-9.
- *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.

Concepts

- *ARM DS-5 Using the Debugger*:
  - *C/C++ editor* on page 11-12
  - *Debug Control view* on page 11-18
  - *Registers view* on page 11-33.

3.10 Loading the calendar application on to a bare metal target

You can use either a DSTREAM or *RealView® ICE* (RVI) connection to load the calendar application on to a bare metal target.

3.10.1 Prerequisites

Before running the calendar application you must:

- Locate and import the calendar application that is provided with the examples as part of the DS-5 installation.
- Connect a debug agent to the host workstation and the target. The debug agent must be powered up and the boot sequence must have completed.
- Obtain the target IP address or name for the connection between the debugger and the debug hardware agent.
- Launch Eclipse.

3.10.2 Procedure

To load an application:

1. Select **Debug Configurations...** from the **Run** menu.
2. Select the `calendar-beagle-example` from the list of DS-5 Debugger configurations.
3. In the Connections panel, enter the IP address or name of the debug agent in the **Debug Hardware Address** field.
4. Click on **Debug** to start the debugger and run to the `main()` function.
5. Debugging requires the DS-5 Debug perspective. If the Confirm Perspective Switch dialog box opens, click on **Yes** to switch perspective.

3.10.3 See also

Tasks

- *Debugging the calendar application using Eclipse* on page 3-20.

Reference

- *DS-5 documentation* on page 4-5
- *ARM DS-5 Using the Debugger*:
  - *Debug Configurations - Connection tab* on page 11-55
  - *Debug Configurations - Files tab* on page 11-58
  - *Debug Configurations - Debugger tab* on page 11-62
  - *Debug Configurations - Environment tab* on page 11-67.

3.11 Debugging the calendar application using Eclipse

This topic describes how to debug the calendar application using Eclipse and a hardware target using a debug agent.

3.11.1 Prerequisites

Before running the calendar application you must:

- Load the calendar application on to a bare metal target.
- Ensure that you are in the DS-5 Debug perspective.
- Ensure that the application is running and stopped at the `main()` function.
- Ensure that the target supports ARM semihosting operations. You can do this by opening the Target view and seaching for the **Allows semihosting** capability. Ensure that it shows `true` in the value column.

3.11.2 Procedure

To debug the calendar application using Eclipse:

1. Set a watchpoint on the global variable `daysInMonth`. You can do this by right-clicking on the variable in the Variables view and selecting **Toggle Watchpoint** from the context menu. The watchpoint is created and listed in the Breakpoints view.
2. Change the properties of the watchpoint to only detect when the watchpoint memory location is written to. You can do this in the Breakpoints view by right-clicking on the watchpoint and selecting **Breakpoint Properties...**. In the type field select **WRITE** and then click **OK**.
3. Click on **Continue** in the Debug Control view toolbar to continue running the application.
4. The debugger presents the App Console view where the application is requesting a date. Enter the following date at the prompt and then press the Return key to complete the request and hand control back to the debugger.

   ```
   2009 11 01
   ```

5. The debugger stops at the watchpoint. Open the Variables view and expand **`Globals`** in the Name column. You can see the global struct `date` and the global variable `daysInMonth`.

   Take note that the number of days in the month is 31. This is incorrect for November. You can correct the source code and rebuild the image but for this particular tutorial, continue with the next step.
6. Change this value in the debugger using one of the following:
   - In the Variables view, enter `30` in the Value column for `daysInMonth`.
   - In the Commands view, enter the following command:

     ```
     set variable "calendar.c"::daysInMonth = 30
     ```
7. Click on **Continue** in the Debug Control view toolbar to continue running the application.
8. Open the App Console view to verify that the debugger displays the correct list of dates.

3.11.3 See also

Tasks

- *Loading the calendar application on to a bare metal target* on page 3-19.

Concepts

- *ARM DS-5 Using the Debugger*:
  - *Debugger concepts* on page 2-3.

Reference

- *ARM DS-5 Debugger Command Reference*:
  - Chapter 2 *DS-5 Debugger commands*.
- *ARM DS-5 Using the Debugger*:
  - *Debug Control view* on page 11-18
  - *Commands view* on page 11-15
  - *App Console view* on page 11-3
  - *Debug Configurations - Connection tab* on page 11-55
  - *Debug Configurations - Files tab* on page 11-58
  - *Debug Configurations - Debugger tab* on page 11-62
  - *Debug Configurations - Arguments tab* on page 11-65
  - *Debug Configurations - Environment tab* on page 11-67.

Chapter 4
DS-5 installation and examples information

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

Reference

4.1 System requirements for DS-5

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 linking or debugging large images, using models with large simulated memory maps, or when using the Streamline Performance Analyzer.

A full installation requires approximately 1 GB of hard disk space.

4.1.1 Supported platforms

DS-5 is supported on the following platforms and service packs:

- Windows XP Professional service pack 3
- Windows Vista Business service pack 2 (deprecated)
- Windows Vista Enterprise service pack 2 (deprecated)
- Windows7 Professional
- Windows 7 Enterprise
- Red Hat Enterprise Linux WS version 5 for Intel x86.

The tools support both 32-bit and 64-bit versions of these operating systems where available, with the exception of *RealView® ICE* (RVI) and DSTREAM which do not support USB connections on 64-bit Linux.

4.1.2 DS-5 Debugger requirements

Before using the debugger for Linux application debugging you must ensure that `gdbserver` is installed on your target. The minimum required version is 6.8.

`gdbserver` executables built for ARMv4T™, ARMv5T™, and Thumb®-2 architectures are provided with DS-5 in *`install_directory`*`\arm`.

The ARM® Linux distribution provided with the DS-5 examples also includes a copy of `gdbserver` in its `cramfs` file system.

The minimum required RVI or DSTREAM firmware version is 4.1. 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

- *Debug hardware configuration utilities* on page 2-8
- *DS-5 installation directories* on page 4-3
- *DS-5 licensing and product updates* on page 4-4.

4.2 DS-5 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

- *DS-5 documentation* on page 4-5
- *DS-5 examples* on page 4-6.

4.3 DS-5 licensing and product updates

DS-5 is a licensed product that uses the FLEX*net* license management software. To request a license or to access the latest DS-5 product information and updates, go to the ARM Self-Service Portal.

4.3.1 See also

Reference

- *FLEXnet for ARM® Tools License Management Guide*, http://infocenter.arm.com/help/topic/com.arm.doc.dui0209-/index.html.

Other information

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

4.4 DS-5 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 the Target Configuration Editor*
- *ARM® DS-5™ Using Eclipse*
- *ARM® DS-5™ Glossary*
- *ARM® Streamline™ Using ARM Streamline*
- *DSTREAM Setting Up the Hardware*
- *DSTREAM System and Interface Design Reference*
- *RealView® ICE and RealView Trace Setting Up the Harware*
- *RealView® ICE and RealView System and Interface Design Reference*
- *DSTREAM and RealView® ICE Using the Debug Hardware Configuration Utilities*

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

- *DS-5 installation directories* on page 4-3
- *DS-5 examples* on page 4-6
- *Documentation on the ARM website*, http://infocenter.arm.com/help/topic/com.arm.doc.subset.swdev.ds5.

4.5 DS-5 examples

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

The code for many of the examples in the documentation is located in a ZIP archive in the examples directory, *examples_directory*\examples.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 launch script that enables you to easily load and debug example code running on a model.

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

Reference

- *DS-5 documentation* on page 4-5
- *DS-5 installation directories* on page 4-3.

Using Eclipse:

- *Using the welcome screen* on page 3-4.

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for panasonic ***manufacturer:***

Other Similar products are found below :

ECE-A1HKAR47 ELK-EA102FA ELC-09D151F EEC-S0HD224H ELL-5PS3R3N HC2-H-DC48V-F HL2-HP-AC120V-F HL2-H-DC12V-F HL2-HP-DC12V-F HL2-HP-DC6V-F HL2-HP-DC24V-F HL2-H-DC110V-F HC4-H-DC24V HL2-HTM-DC24V-F HL2-HTM-AC24V-F HC4-H-AC24V HC4-H-AC120V HC4-H-DC12V EEC-RG0V155H AZH2031 RP-SDMF64DA1 EEF-UD0K101R EVM-F6SA00B55 RP-SMLE08DA1 ELC-12D101E ERA-3YEB272V EEC-RF0V684 ERA-3YEB153V ELC-3FN2R2N ERA-3YEB512V ERJ-1GEJ564C ERZ-V20R391 ETQ-P3W3R3WFN ELL-ATV681M ELK-EA100FA EEF-UD0J101R LC-R121R3P ERA-3YEB303V ERZ-V05V680CB EEF-UE0K101R ELK-E101FA EEC-S0HD224V EVQ-PAC05R ELK-EA222FA LT4H-DC24V LT4HL8-AC24V LT4HW-AC24V LT4HWT8-AC240V LT4HWT-AC240VS CX-444-P-Z