# ARM® DS-5

**Version 5.27**

## Getting Started Guide

ARM


ARM 100950_0527_00_en

# ARM® DS-5

## Getting Started Guide



### Release Information

### Document History

| Issue | Date | Confidentiality | Change |
|---|---|---|---|
| A | 30 June 2010 | Non-Confidential | First release |
| B | 30 September 2010 | Non-Confidential | Update for DS-5 version 5.2 |
| C | 30 November 2010 | Non-Confidential | Update for DS-5 version 5.3 |
| D | 30 January 2011 | Non-Confidential | Update for DS-5 version 5.4 |
| F | 30 July 2011 | Non-Confidential | Update for DS-5 version 5.6 |
| G | 30 September 2011 | Non-Confidential | Update for DS-5 version 5.7 |
| H | 30 November 2012 | Non-Confidential | Update for DS-5 version 5.8 |
| I | 28 February 2012 | Non-Confidential | Update for DS-5 version 5.9 |
| J | 30 May 2012 | Non-Confidential | Update for DS-5 version 5.10 |
| K | 30 July 2012 | Non-Confidential | Update for DS-5 version 5.11 |
| L | 30 October 2012 | Non-Confidential | Update for DS-5 version 5.12 |
| M | 15 December 2012 | Non-Confidential | Update for DS-5 version 5.13 |
| N | 15 March 2013 | Non-Confidential | Update for DS-5 version 5.14 |
| O | 14 June 2013 | Non-Confidential | Update for DS-5 version 5.15 |
| P | 13 September 2013 | Non-Confidential | Update for DS-5 version 5.16 |
| Q | 13 December 2013 | Non-Confidential | Update for DS-5 version 5.17 |
| R | 14 March 2014 | Non-Confidential | Update for DS-5 version 5.18 |
| S | 27 June 2014 | Non-Confidential | Update for DS-5 version 5.19 |
| T | 17 October 2014 | Non-Confidential | Update for DS-5 version 5.20 |
| U | 20 March 2015 | Non-Confidential | Update for DS-5 version 5.21 |
| V | 15 July 2015 | Non-Confidential | Update for DS-5 version 5.22 |
| W | 15 October 2015 | Non-Confidential | Update for DS-5 version 5.23 |
| X | 15 March 2016 | Non-Confidential | Update for DS-5 version 5.24 |
| Y | 15 July 2016 | Non-Confidential | Update for DS-5 version 5.25 |
| Z | 18 November 2016 | Non-Confidential | Update for DS-5 version 5.26 |
| 0527-00 | 07 April 2017 | Non-Confidential | Document numbering scheme has changed. Update for DS-5 version 5.27. |

### Non-Confidential Proprietary Notice

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

### 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.

Unrestricted Access is an ARM internal classification.

### 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 Guide


**Preface**

About this book ..... 10

**Chapter 1 ARM® DS-5 Product Overview**

1.1 About DS-5 ..... 1-13
1.2 About Eclipse for DS-5 ..... 1-14
1.3 About DS-5 Debugger ..... 1-15
1.4 About Fixed Virtual Platform (FVP) ..... 1-16
1.5 About ARM® Compiler tools ..... 1-17
1.6 About ARM® Streamline Performance Analyzer ..... 1-19
1.7 Debug options supported by DS-5 ..... 1-20
1.8 About debug hardware configuration ..... 1-22

**Chapter 2 ARM® DS-5 installation and system requirements**

2.1 System requirements ..... 2-24
2.2 Installing DS-5 ..... 2-26
2.3 Installation directories ..... 2-28

**Chapter 3 Licensing ARM® DS-5**

3.1 Licensing and product updates ..... 3-30
3.2 Viewing and editing licenses using the ARM License Manager ..... 3-31
3.3 Using a serial number or activation code to obtain a license ..... 3-33
3.4 Using an existing license file or license server to obtain a license ..... 3-35
3.5 Evaluating DS-5 Ultimate Edition ..... 3-36

3.6 Obtaining a license manually using the ARM website ..... 3-38
3.7 Deleting a license ..... 3-40
3.8 Changing the Toolkit ..... 3-41
3.9 Viewing detailed license and system information ..... 3-42

**Chapter 4 Working with ARM® DS-5**

4.1 Documentation provided with DS-5 ..... 4-45
4.2 Examples provided with DS-5 ..... 4-46
4.3 Importing the example projects into Eclipse ..... 4-48
4.4 Creating a C or C++ project ..... 4-51
4.5 Setting up the compilation tools for a specific build configuration ..... 4-53
4.6 Building the project ..... 4-55
4.7 Creating a new DS-5 debug configuration for an FVP connection ..... 4-56
4.8 Running an application on an FVP with varying vector width ..... 4-60
4.9 Building the gnometris project from Eclipse ..... 4-64
4.10 Building the gnometris project from the command line ..... 4-65
4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP) ..... 4-66
4.12 Loading the Gnometris application on to an ARM® Linux target ..... 4-67
4.13 Configuring an RSE connection to work with an ARM® Linux target ..... 4-68
4.14 Launching gdbserver with an application ..... 4-74
4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target ..... 4-75
4.16 Debugging Gnometris ..... 4-78
4.17 Debugging a loadable kernel module ..... 4-79
4.18 Performance analysis of the threads application running on ARM® Linux ..... 4-83
4.19 About registering a new compiler toolchain ..... 4-85
4.20 Registering a compiler toolchain from the DS-5 command prompt ..... 4-86
4.21 Configuring a compiler toolchain for the DS-5 command prompt ..... 4-90
4.22 Registering a compiler toolchain from Eclipse ..... 4-91

# List of Figures
# ARM® DS-5 Getting Started Guide


*Figure 4-1 Import DS-5 Examples and Programming Libraries ..... 4-48*
*Figure 4-2 Select DS-5 Examples and Programming Libraries ..... 4-49*
*Figure 4-3 Creating a new C project ..... 4-51*
*Figure 4-4 Typical build settings dialog box for a C project ..... 4-54*
*Figure 4-5 Compiled axf file ..... 4-55*
*Figure 4-6 Create a new debug configuration ..... 4-56*
*Figure 4-7 Select an FVP model to connect to ..... 4-57*
*Figure 4-8 Specify the application to load ..... 4-58*
*Figure 4-9 Specify the debug symbol ..... 4-59*
*Figure 4-10 Debug Control view ..... 4-59*
*Figure 4-11 Model parameters for vector length 128 bits (veclen=2) ..... 4-60*
*Figure 4-12 SVE registers with vector length 128 bits (veclen=2) ..... 4-61*
*Figure 4-13 Target console with vector length 128 bits (veclen=2) ..... 4-61*
*Figure 4-14 Model parameters for vector length 2048 bits (veclen=32) ..... 4-62*
*Figure 4-15 SVE registers with vector length 2048 bits (veclen=32) ..... 4-62*
*Figure 4-16 Target console with vector length 2048 bits (veclen=32) ..... 4-63*
*Figure 4-17 Selecting a connection type ..... 4-68*
*Figure 4-18 Enter connection information ..... 4-69*
*Figure 4-19 Sftp Files options ..... 4-70*
*Figure 4-20 Defining the shell services ..... 4-71*
*Figure 4-21 Defining the terminal services ..... 4-72*
*Figure 4-22 Typical connection configuration for Linux application debug ..... 4-75*
*Figure 4-23 Typical file selection for Linux application debug ..... 4-76*

Figure 4-24 Typical debugger settings for Linux application debug ..... 4-77
Figure 4-25 Typical connection settings for a Linux kernel/Device Driver Debug ..... 4-80
Figure 4-26 Typical Files settings for a Linux kernel/Device Driver Debug ..... 4-81
Figure 4-27 Streamline Data view ..... 4-83
Figure 4-28 Streamline analysis report for the threads application ..... 4-84
Figure 4-29 Registering a new toolchain ..... 4-86
Figure 4-30 Registering a new toolchain ..... 4-87
Figure 4-31 Using a new toolchain for a new project ..... 4-88
Figure 4-32 Changing the toolchain for a project ..... 4-89
Figure 4-33 Configuring a default toolchain ..... 4-90
Figure 4-34 Toolchains Preferences dialog ..... 4-91
Figure 4-35 Properties for the new toolchain ..... 4-92

# List of Tables
# ARM® DS-5 Getting Started Guide


*Table 1-1* *ARM Compiler Tools* ..... *1-17*
*Table 2-1* *DS-5 default directories* ..... *2-28*

# Preface

This preface introduces the *ARM® DS-5 Getting Started Guide*.

It contains the following:

- *About this book* on page 10.

# About this book

ARM DS-5 Getting Started Guide describes the installation and system requirements. It also explains how to work with DS-5.

## Using this book

This book is organized into the following chapters:

### *Chapter 1 ARM® DS-5 Product Overview*

Gives an overview of the main features of ARM® DS-5.

### *Chapter 2 ARM® DS-5 installation and system requirements*

This chapter provides information on the installation and system requirements for ARM DS-5.

### *Chapter 3 Licensing ARM® DS-5*

Describes how to manage DS-5 licenses using the ARM License Manager within the Eclipse environment.

### *Chapter 4 Working with ARM® DS-5*

This chapter explains how to run and debug applications using ARM DS-5 tools. It also provides information about the examples and documentation provided with DS-5.

## Glossary

The ARM Glossary is a list of terms used in ARM documentation, together with definitions for those terms. The ARM Glossary does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See the *ARM Glossary* for more information.

## Typographic conventions

*italic*
: Introduces special terminology, denotes cross-references, and citations.

**bold**
: Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

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

`mono`space
: Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

*`monospace italic`*
: Denotes arguments to monospace text where the argument is to be replaced by a specific value.

**`monospace bold`**
: Denotes language keywords when used outside example code.

`<and>`
: Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

## Feedback

### Feedback on this product

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

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

### Feedback on content

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

- The title *ARM DS-5 Getting Started Guide*.
- The number ARM 100950_0527_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

**Note**

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

## Other information

- *ARM Developer*.
- *ARM Information Center*.
- *ARM Technical Support Knowledge Articles*.
- *Support and Maintenance*.
- *ARM Glossary*.

# Chapter 1
# ARM® DS-5 Product Overview

Gives an overview of the main features of ARM® DS-5.

It contains the following sections:


- *1.1 About DS-5* on page 1-13.
- *1.2 About Eclipse for DS-5* on page 1-14.
- *1.3 About DS-5 Debugger* on page 1-15.
- *1.4 About Fixed Virtual Platform (FVP)* on page 1-16.
- *1.5 About ARM® Compiler tools* on page 1-17.
- *1.6 About ARM® Streamline Performance Analyzer* on page 1-19.
- *1.7 Debug options supported by DS-5* on page 1-20.
- *1.8 About debug hardware configuration* on page 1-22.

## 1.1 About DS-5

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

It includes:

- 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.
- DS-5 Debugger, a graphical debugger supporting software development on ARM processor-based targets and *Fixed Virtual Platform* (FVP) targets.
- *Fixed Virtual Platform* (FVP) targets enable development of software without the requirement for actual hardware.
- ARM Streamline is a graphical performance analysis tool that enables you to transform sampling data and system trace into reports that present the data in both visual and statistical forms.
- ARM Compiler 5 and ARM Compiler 6 toolchains enable you to build embedded and bare-metal code.
- You can use the debug hardware firmware configuration views in DS-5 to update and configure the debug hardware unit that provides the interface between your development platform and your workstation.
- Dedicated examples, applications, and supporting documentation to help you get started with using the DS-5 tools.

Some third-party compilers are compatible with DS-5. For example, the GNU Compiler tools enable you to compile bare-metal, Linux kernel, and Linux applications for ARM targets.

### Related concepts

*1.2 About Eclipse for DS-5* on page 1-14.
*1.3 About DS-5 Debugger* on page 1-15.
*1.4 About Fixed Virtual Platform (FVP)* on page 1-16.
*1.5 About ARM® Compiler tools* on page 1-17.
*1.6 About ARM® Streamline Performance Analyzer* on page 1-19.
*1.8 About debug hardware configuration* on page 1-22.

### Related references

*3.1 Licensing and product updates* on page 3-30.
*4.1 Documentation provided with DS-5* on page 4-45.
*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*DS-5 Knowledge Articles*.

## 1.2 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 includes:

**Project manager**
: The project manager 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**
: Editors enable you to read, write, or modify C/C++ or ARM assembly language source files.

**Perspectives and views**
: Perspectives provide customized views, menus, and toolbars to suit a particular type of environment. DS-5 uses the **C/C++**, **DS-5 Debug**, and **DS-5 Configuration** perspectives. To switch perspectives, from the main menu, select **Window** > **Open Perspective**.

### Related concepts

*1.1 About DS-5* on page 1-13.

### Related information

*Getting started with Eclipse.*

## 1.3 About DS-5 Debugger

DS-5 Debugger, a graphical debugger supporting software development on ARM processor-based targets and *Fixed Virtual Platform* (FVP) targets.

It makes it easy to debug bare-metal and Linux 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 applications 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.

### Related concepts

*1.1 About DS-5* on page 1-13.
*1.4 About Fixed Virtual Platform (FVP)* on page 1-16.

### Related information

*Getting started with the debugger.*

## 1.4 About *Fixed Virtual Platform* (FVP)

*Fixed Virtual Platform* (FVP) targets enable development of software without the requirement for actual hardware. The functional behavior of an *Fixed Virtual Platform* (FVP) is equivalent to real hardware from a programmers view.

When using an *Fixed Virtual Platform* (FVP), 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 provides *several FVPs*, covering a range of processors in the Cortex® family. You can also connect to a variety of other ARM and third-party simulation models implementing CADI.

DS-5 provides an ARMv8-A Fixed Virtual Platform (FVP) executable that supports the SVE architecture extension. You can also connect to a variety of other ARM and third-party simulation models.

The executables are located in `tools_directory`. You can use them to run your applications from either the command-line or within Eclipse. See *2.3 Installation directories* on page 2-28 for more information about various directories that are installed with DS-5.

### Related concepts

*1.1 About DS-5* on page 1-13.

### Related references

*2.3 Installation directories* on page 2-28.

### Related information

*Fixed Virtual Platforms FVP Reference Guide.*

## 1.5 About ARM® Compiler tools

ARM Compiler tools enable you to build applications and libraries suitable for bare-metal embedded systems.

DS-5 provides two versions of ARM Compiler for compiling embedded and bare-metal applications:

- ARM Compiler 5 - Supports all ARM architectures from ARMv4 to ARMv7 inclusive.

  ——— **Note** ———

  All architectures before ARMv4 are obsolete and are no longer supported by ARM Compiler 5.

  ———————————

- ARM Compiler 6 - Supports ARMv6-M, ARMv7, and ARMv8 architectures. ARM Compiler 6 also supports optimization for the SVE architectural extensions. This toolchain is recommended for software development targeting ARMv8-A with the SVE architectural extensions for High Performance Computing (HPC).

The ARM Compiler tools are located in the *`ARM Compiler 5`* and *`ARM Compiler 6`* directories within the DS-5 installation. You can use them to build your applications from either the command-line or within Eclipse.

**Table 1-1 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` | ARM Compiler 5. This compiles your C and C++code. It supports inline and embedded assemblers. |
| `armclang` | ARM Compiler 6 and Assembler. This compiles C and C++ code, and assembles A32, A64, and T32 GNU syntax assembly code. The `armclang` compiler also contains an advanced auto-vectorizer, capable of taking advantage of SVE features. `armclang` is also able to compile assembly files containing SVE instructions. It supports inline GNU syntax assembly code. |
| `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. |
| `llvm-objdump` | Use the `llvm-objdump` tool with the SVE target feature enabled to display the details and contents of an ELF-format binary file. This includes disassembly of the text section of an object containing SVE instructions. |

### ARM® Compiler Scalable Vector Extension User Guide

The *ARM® Compiler Scalable Vector Extension User Guide* contains additional information about:

- Assembling SVE code.
- Disassembling SVE object files.
- Compiling C and C++ code for SVE-enabled targets.
- Running a binary in an ARMv8-A Fixed Virtual Prototype (FVP).
- Best practices to enable auto-vectorization.
- Auto-vectorization examples.
- Embedding SVE assembly code directly into C and C++ code.
- Using pragmas to encourage or suppress auto-vectorization.

- Various compiler options.
- General troubleshooting advice.

To view the *ARM® Compiler Scalable Vector Extension User Guide*, from the main DS-5 menu, click **Help** > **Help Contents**. In the help window, browse and select **ARM DS-5 Documentation**. The document is available under **ARM Compiler 6 Help Contents** > **User information**.

**Note**

ARM Compiler is license managed. Specific features depend on your installed license.

For example, a license might limit the use of ARM Compiler to specific processor types, or place a maximum limit on the size of images that can be produced, or require that you work with proprietary format (ORC) objects instead of ELF format objects.

You can enable additional features by purchasing a license for the full DS-5 suite. Contact your tools supplier for details.

### Related concepts

*1.1 About DS-5* on page 1-13.

*4.19 About registering a new compiler toolchain* on page 4-85.

### Related tasks

*4.20 Registering a compiler toolchain from the DS-5 command prompt* on page 4-86.

### Related references

*2.3 Installation directories* on page 2-28.

### Related information

*Creating a new C or C++ project.*

## 1.6 About ARM® Streamline Performance Analyzer

ARM Streamline is a graphical performance analysis tool that enables you to transform 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.

ARM Streamline does not support the SVE architectural extensions to ARMv8-A.

### Related concepts

*1.1 About DS-5* on page 1-13.

### Related tasks

*4.18 Performance analysis of the threads application running on ARM® Linux* on page 4-83.

### Related information

*Streamline User Guide.*

## 1.7 Debug options supported by DS-5

DS-5 supports various debug options.

Debug adapters vary in complexity and capability but, combined with software debug agents, they provide high-level debug functionality for the target that is being debugged, for example:

- Reading/Writing registers.
- Setting breakpoints.
- Reading from memory.
- Writing to memory.

**Note**

A debug adapter or connection is not the application being debugged, nor the debugger itself.

Supported ARM debug hardware adapters include:

- ARM DSTREAM.

  **Note**

  You must use DSTREAM for ARMv8 development.

- ARM RVI™.
- Keil® ULINK™2.
- Keil® ULINK™pro.
- Keil® ULINK™pro D.

Supported debug connections include:

- CADI (debug interface for models).
- Ethernet to gdbserver.
- ARM VSTREAM.
- CMSIS-DAP.
- Undodb-server for Linux application rewind.
- Yokogawa Digital Computer Corporation adviceLUNA (JTAG ICE).
- Altera USB-Blaster II.

  **Note**

  DS-5 Debugger can connect to Altera Arria V SoC and Cyclone V SoC boards using Altera USB-Blaster and USB-Blaster II debug units.

  To enable the connections, ensure that the environment variable *`QUARTUS_ROOTDIR`* is set and contains the path to the Altera Quartus tools installation.

  On Windows, this environment variable is usually set by the Quartus tools installer. On Linux, you might have to manually set the environment variable to the Altera Quartus tools installation path. For example, `~/altera/13.0/qprogrammer`.

  For information on installing device drivers for USB-Blaster and USB-Blaster II, consult your Altera Quartus tools documentation.

**Note**

There is currently no hardware available for the SVE architecture extension. Debugging software for the SVE architecture extension is only supported on the ARMv8-A FVP model with the SVE plugin provided with DS-5.

### Related information

*Setting up the ARM DSTREAM Hardware.*

## 1.8 About debug hardware configuration

You can use the debug hardware firmware configuration views in DS-5 to update and configure the debug hardware unit that provides the interface between your development platform and your workstation.

The following views are provided:

**Debug Hardware Config IP view**
Used to *configure the IP address* on a debug hardware unit.

**Debug Hardware Firmware Installer view**
Used to *update the firmware* on a debug hardware unit.

### Related concepts

*1.1 About DS-5* on page 1-13.

# Chapter 2
# ARM® DS-5 installation and system requirements

This chapter provides information on the installation and system requirements for ARM DS-5.

It contains the following sections:


- *2.1 System requirements* on page 2-24.
- *2.2 Installing DS-5* on page 2-26.
- *2.3 Installation directories* on page 2-28.

## 2.1 System requirements

To install and use DS-5, your workstation must have a minimum specification of a dual core 2GHz processor (or equivalent) and 2GB of RAM.

To improve performance when debugging large images, using models with large simulated memory maps, or when using ARM Streamline Performance Analyzer, 4GB of RAM, or more is recommended.

A full installation also requires approximately 3GB of hard disk space.

### Host platform requirements

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

- Windows 10
- Windows 7 Professional Service Pack 1
- Windows 7 Enterprise Service Pack 1
- Red Hat Enterprise Linux 6 Workstation
- Red Hat Enterprise Linux 7 Workstation
- Ubuntu Desktop Edition 12.04 LTS
- Ubuntu Desktop Edition 14.04 LTS

——— **Note** ———

DS-5 is supported on 64-bit host platforms only.

———————————

In addition, ARM Compiler toolchains are supported on the following host platforms:

- Windows 8.1 (64-bit only) (ARM Compiler 5 and 6 toolchains only)
- Windows Server 2012 (64-bit only) (ARM Compiler 5 and 6 toolchains only)

### Debug system requirements

Linux application debug requires `gdbserver` on your target. The recommended version of `gdbserver` is 7.0 or later.

——— **Note** ———

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

———————————

Linux application rewind requires *`undodb-server`* on your target. DS-5 Debugger copies *`undodb-server`* to the target for you in the **Download and Debug** connection type, but for all other connection types, you must copy it yourself. The *`undodb-server`* binary is located in the *`DS-5_install_directory`*`\arm\undodb\linux` directory within your installation.

DS-5 support for Linux application debug depends on infrastructure and features that are introduced in specific kernel versions:

- DS-5 Debugger supports debugging ARM Linux kernel versions 2.6.28 and later.
- Application debug on *Symmetric MultiProcessing* (SMP) systems requires ARM Linux kernel version 2.6.36 or later.
- Access to VFP and NEON registers require ARM Linux kernel version 2.6.30 or later and `gdbserver` version 7.0 or later.
- ARM Streamline Performance Analyzer supports ARM Linux kernel versions 3.4 and later.

### Additional tools for Linux kernel and bare-metal hardware debugging

ARM Linux kernel and bare-metal debugging require the use of additional tools (not supplied with DS-5) to connect to your hardware target system.

DSTREAM, RVI, ULINKpro, ULINKpro D, and ULINK2 debug units enable connection to physical hardware targets.

**Note**

You must use DSTREAM for ARMv8 development.

VSTREAM enables connection to RTL simulators and hardware emulators.

### Managing firmware updates

- For DSTREAM and RVI use the ***Debug Hardware Firmware Installer*** view to check the firmware and update it if necessary. Updated firmware is available in the *DS-5_install_directory*`/sw/debughw/firmware` directory.
- For VSTREAM, the firmware is delivered as part of the VSTREAM software. To update the firmware, you must install a newer version of VSTREAM.
- For ULINK2 target connection probe to work with DS-5 Debugger, it must be upgraded with CMSIS-DAP compatible firmware. The `UL2_Upgrade.exe` program (Windows only) can upgrade your ULINK2 unit for you. The program and instructions are available in the *DS-5_install_directory*`/sw/debughw/ULINK2` directory.
- For ULINKpro and ULINKpro D, DS-5 manages the firmware.

### Related references

*2.3 Installation directories* on page 2-28.

*3.1 Licensing and product updates* on page 3-30.

*4.1 Documentation provided with DS-5* on page 4-45.

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*Setting up the ARM DSTREAM Hardware.*

*Setting up the ARM RVI Hardware.*

*DS-5 Knowledge Articles.*

*Adobe Viewer.*

## 2.2 Installing DS-5

DS-5 64-bit install packages are available for Windows and Linux platforms.

The main advantage of using a 64-bit version of DS-5 is that the binaries provided with 64-bit versions are capable of processing larger data sets before hitting per-process memory limits. On Linux, 64-bit tools have fewer operating system compatibility issues.

### Installing on Linux

To install DS-5 on Linux, run (not source) `install.sh` and follow the on-screen instructions.

Installing device drivers and desktop shortcuts is optional. The device drivers allow USB connection to debug hardware units, for example DSTREAM. The desktop menu is created using the *http://www.freedesktop.org/* menu system on supported Linux platforms. If you want to install these features post-install, using root privileges, run `run_post_install_for_ARM_DS-5.sh` script available in the install directory.

**Note**

Tools installed by the 64-bit installer have dependencies on 32-bit system libraries.

You must ensure that 32-bit compatibility libraries are installed when using DS-5 on 64-bit Linux host platforms. DS-5 tools may fail to run or report errors about missing libraries if 32-bit compatibility libraries are not installed.

There are known issues when running DS-5 32-bit binaries on 64-bit Ubuntu host platforms.

The ARM Knowledgebase contains information which may help you troubleshoot these issues: *http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka14522.html*

**Note**

On Linux, you can use `suite_exec` to configure the environment variables correctly for DS-5. For example, run *`DS-5_install_directory`*`/bin/suite_exec <shell>` to open a shell with the PATH and other environment variables correctly configured. Run `suite_exec` with no arguments for more help.

### Installing on Windows

To install DS-5 on Windows, run `setup.exe` and follow the on-screen instructions.

During installation, you may be prompted to install device drivers. These drivers allow USB connections to DSTREAM, RVI, and Energy Probe hardware units. They also support networking for the simulation models. It is recommended to install these drivers if you intend to use these features.

**Note**

During installation, you might receive warnings about driver software. You can safely ignore warnings displayed when these drivers are installed and continue with the installation.

### Command-line installation on Windows

Command-line installation and uninstallation are possible on Windows by opening a command prompt, with administrative privileges, and running Microsoft's installer, `msiexec.exe`. You must provide the location of the `.msi` file as an argument to `msiexec`. You can get a full list of options for using `msiexec` by running `msiexec /?` on the command-line. An example of how to install DS-5 using `msiexec` is:

`msiexec.exe /i` *`installer_location\data\install.msi`* `EULA=1 /qn /l*v` *`install.log`*

Where:

`/i`
: This option is to perform the installation.

*`installer_location\data\install.msi`*
: This specifies the full pathname of the `.msi` file to install.

`/EULA=1`
: This is an ARM specific option. Setting EULA to 1 means you accept the End User License Agreement (EULA). You must read the EULA in the GUI installer before accepting it on the command-line.

`/qn`
: This option specifies quiet mode, so that the installation does not require user interaction.

  **Note**

  Device driver installation still requires user interaction. If you do not require USB drivers or if you want the installation to avoid user interaction for USB drivers, you can use `SKIP_DRIVERS=1` option on the command-line.

`/l*v` *`install.log`*
: This option specifies the log file to log all output from the installation.

### Installing multiple versions of DS-5

You can install multiple versions of DS-5 on Windows and Linux platforms.

You can select different toolkits for different installations of DS-5. The default installation of DS-5 does not automatically select a toolkit. You must select the appropriate toolkit in each installation of DS-5.

### Related tasks

*3.8 Changing the Toolkit* on page 3-41.

## 2.3 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 *DS-5_install_directory* shown is the default installation directory. The DS-5 version number, *<version>*, is part of the default installation directory name. If you installed the product in a different directory, then the path names are relative to your chosen directory.

**Table 2-1 DS-5 default directories**

| Directory | Windows | Linux |
|---|---|---|
| *DS-5_install_directory* | C:\Program Files\DS-5 v*<version>* | ~/DS-5_v*<version>* |
| *arm_directory* | *DS-5_install_directory*\arm\... | *DS-5_install_directory*/arm/... |
| *examples_directory* | *DS-5_install_directory*\examples\... | *DS-5_install_directory*/examples/... |
| *tools_directory* | *DS-5_install_directory*\bin\... | *DS-5_install_directory*/bin/... |
| *documents_directory* | *DS-5_install_directory*\documents\... | *DS-5_install_directory*/documents/... |

### Related references

*2.1 System requirements* on page 2-24.
*3.1 Licensing and product updates* on page 3-30.
*4.1 Documentation provided with DS-5* on page 4-45.
*4.2 Examples provided with DS-5* on page 4-46.

# Chapter 3
# Licensing ARM® DS-5

Describes how to manage DS-5 licenses using the ARM License Manager within the Eclipse environment.

It contains the following sections:


- *3.1 Licensing and product updates* on page 3-30.
- *3.2 Viewing and editing licenses using the ARM License Manager* on page 3-31.
- *3.3 Using a serial number or activation code to obtain a license* on page 3-33.
- *3.4 Using an existing license file or license server to obtain a license* on page 3-35.
- *3.5 Evaluating DS-5 Ultimate Edition* on page 3-36.
- *3.6 Obtaining a license manually using the ARM website* on page 3-38.
- *3.7 Deleting a license* on page 3-40.
- *3.8 Changing the Toolkit* on page 3-41.
- *3.9 Viewing detailed license and system information* on page 3-42.

## 3.1 Licensing and product updates

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

When you first launch DS-5, the ARM License Manager might report that there is no license configured. You must configure a valid license in the ARM License Manager.

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

To compare DS-5 editions, see: *Which version of ARM DS-5 Development Studio is right for me?*

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

### Related tasks

*3.8 Changing the Toolkit* on page 3-41.

### Related references

*2.1 System requirements* on page 2-24.

*2.3 Installation directories* on page 2-28.

*4.1 Documentation provided with DS-5* on page 4-45.

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*ARM Forums.*

*ARM DS-5 License Management Guide.*

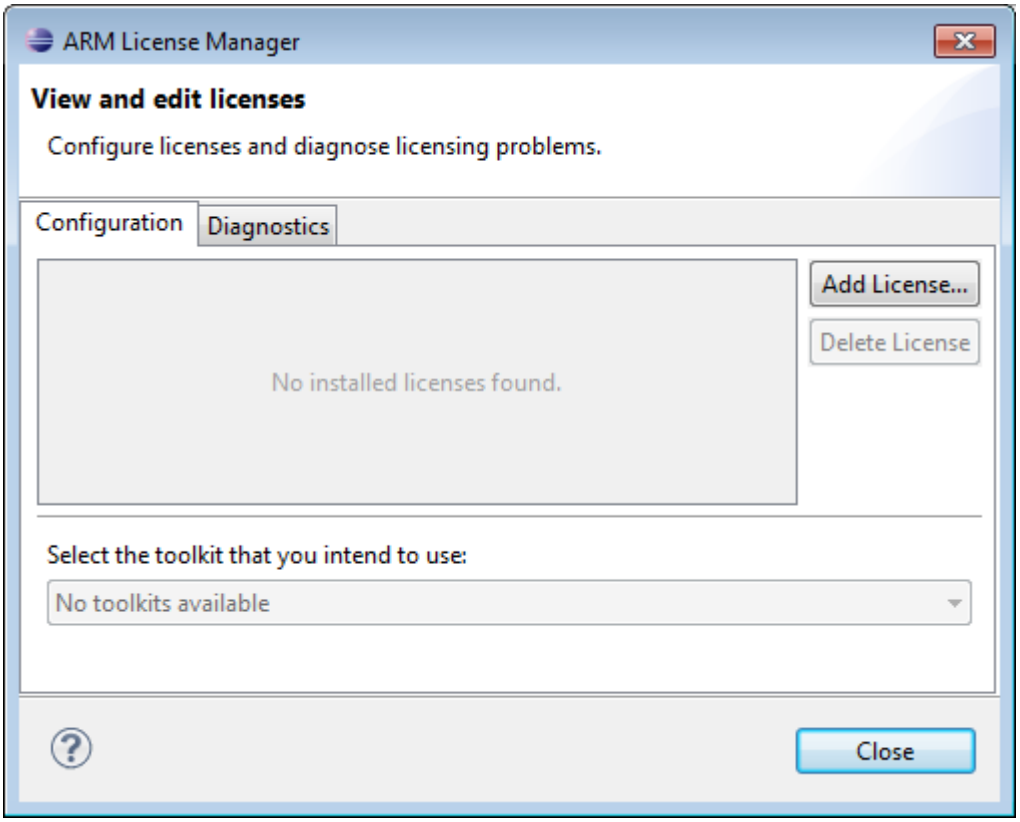
*ARM Self-Service Portal.*

## 3.2 Viewing and editing licenses using the ARM License Manager

You can view and edit DS-5 licenses using the **ARM License Manager**.

### Procedure

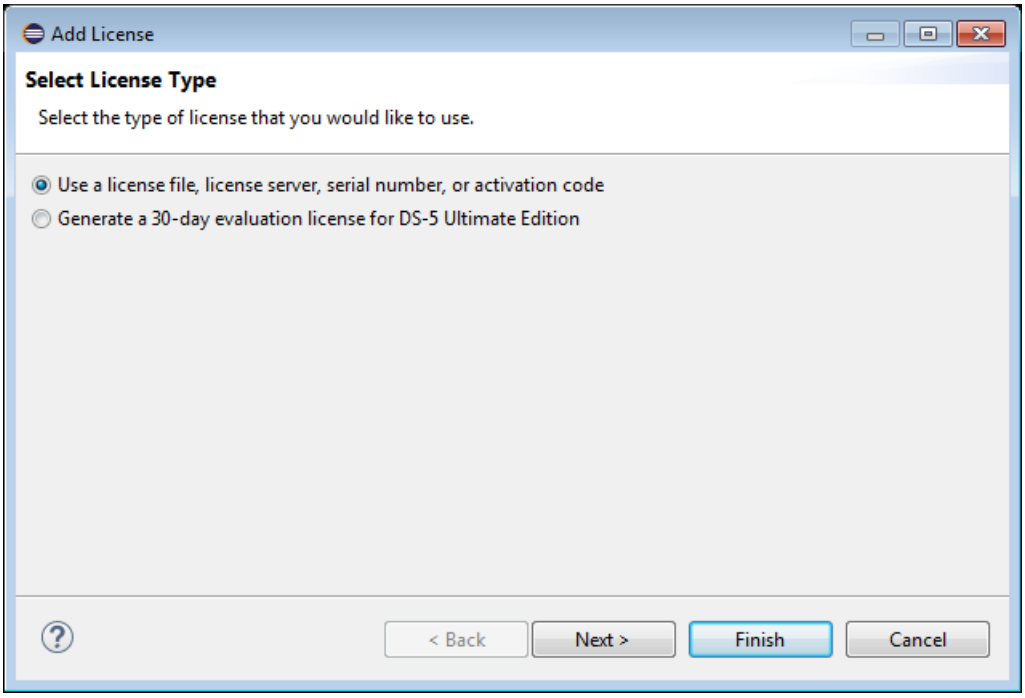
1. To view the **ARM License Manager**, in Eclipse, select **Help** > **ARM License Manager...**.




**Note**

Installed licenses are displayed in the **Configuration** tab of the ARM License Manager dialog box.

2. To add a license to DS-5, click **Add License...**. Use the options in the Add License dialog box to obtain a new license. You must first select the license type.

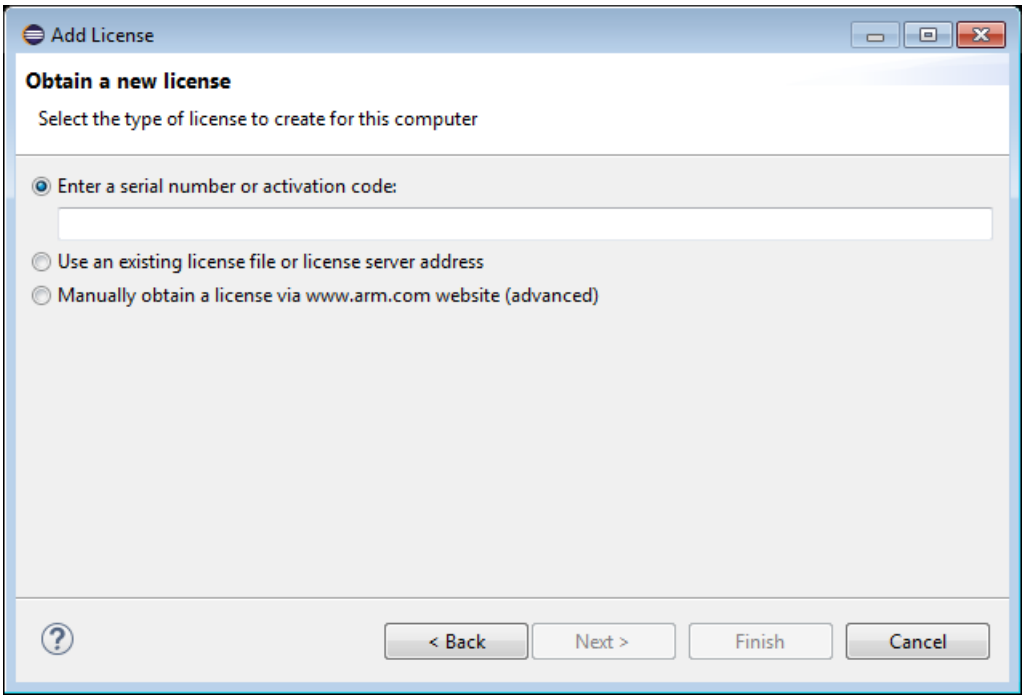


## 3.3 Using a serial number or activation code to obtain a license

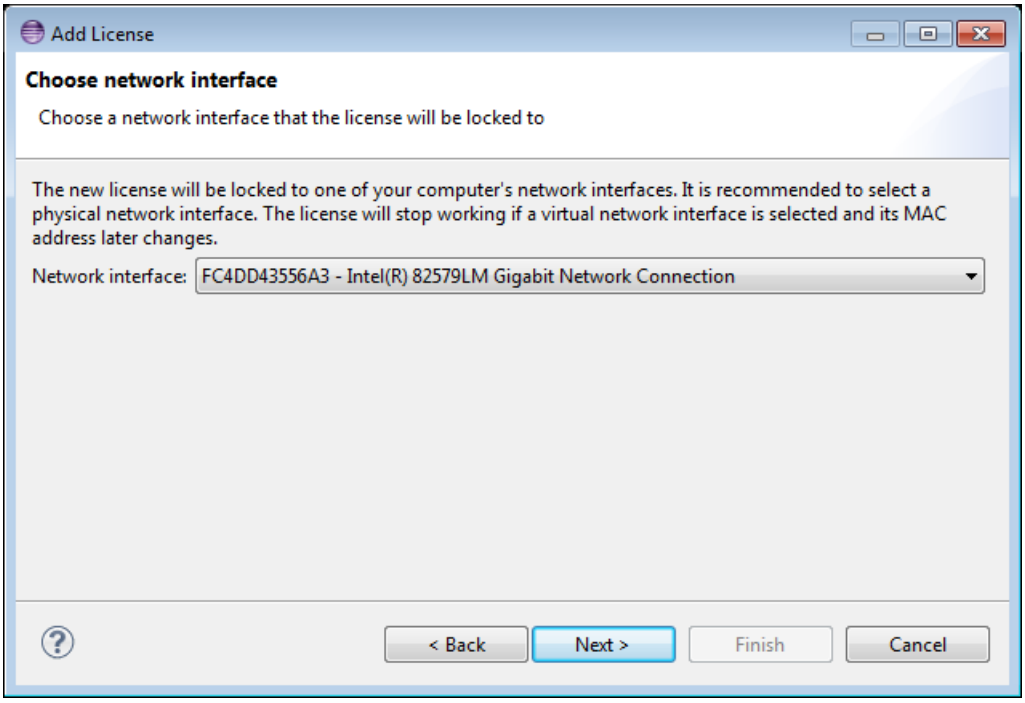
You can use a serial number or activation code to obtain a license.

### Procedure

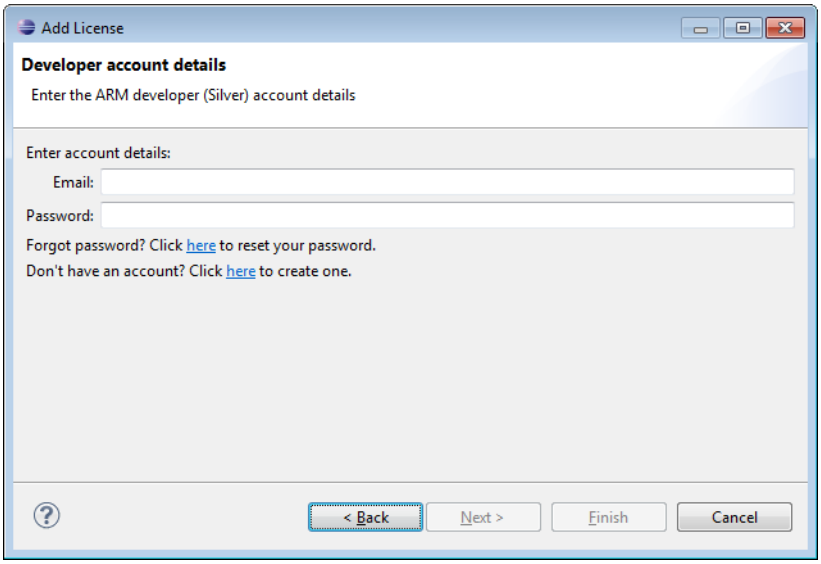
1. In the **Add License** dialog box, select **Use a license file, license server, serial number, or activation code**. Then click **Next**.
2. In the **Obtain a new license** page, select **Enter a serial number or activation code**.




3. Enter the serial number in the field. Then click **Next**
4. In the **Choose network interface** dialog, select a **Network interface** from the drop-down list.




5. Click **Next**.
6. Enter your ARM developer account details in the ARM Self-Service Portal or if you do not have an account, you can create one.



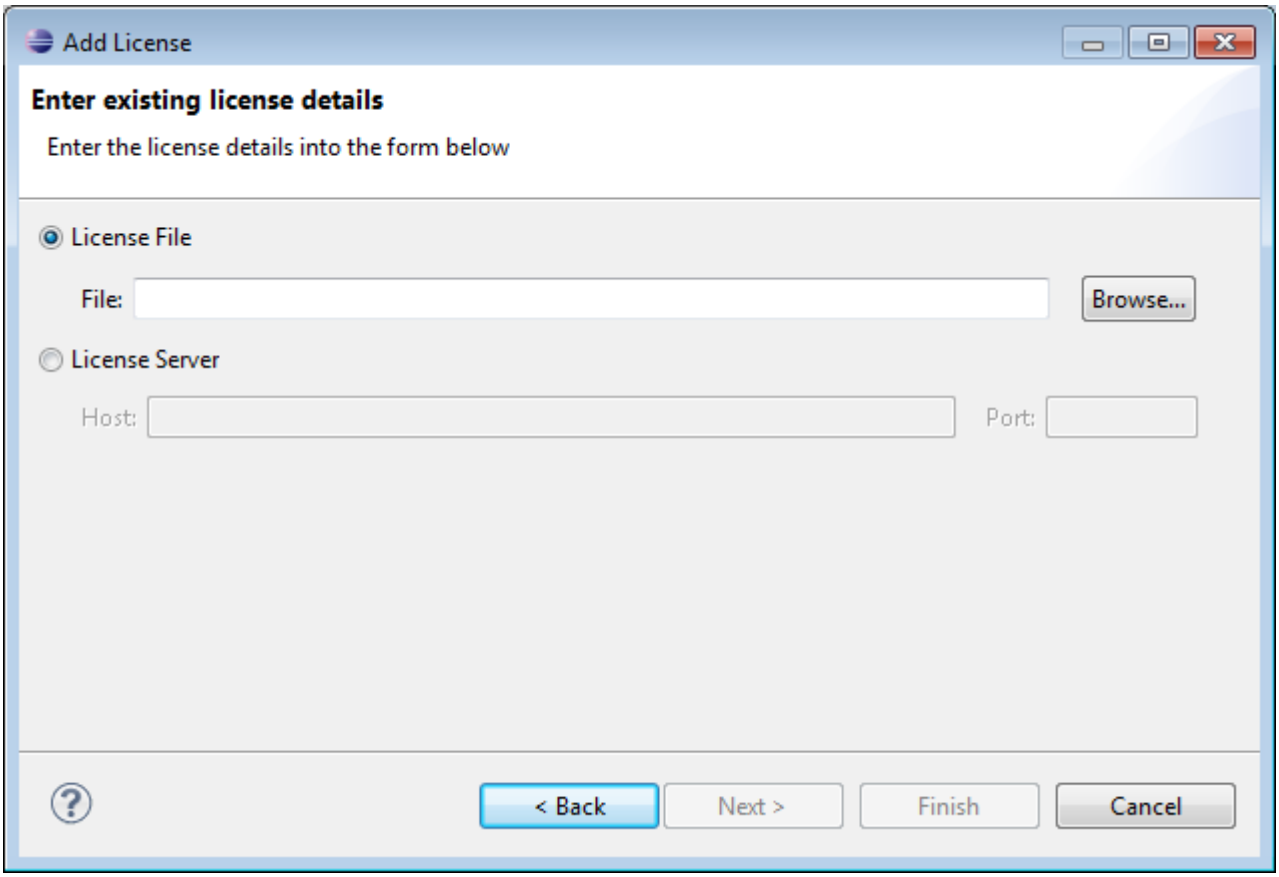

7. Click **Finish**.

## 3.4 Using an existing license file or license server to obtain a license

You can obtain a license using an existing license file or license server.

### Procedure

1. In the **Add License** dialog box, select **Use a license file, license server, serial number, or activation code**. Then click **Next**.
2. In the **Obtain a new license** page, select **Use an existing license file or license server address**. Then click **Next**.
3. In the Enter existing license details dialog, if you have a license file, select **License File** or if you have a server to administer the license, select **License Server**.




**Note**

For server licenses, instead of entering the host and port information separately in their respective fields, you can enter them in the format `port@host` in the Host field.

4. Click **Finish** to add the license to the ARM License Manager.
   In Windows, license files are copied into the `%APPDATA%\ARM\DS-5\licenses` folder. In Linux, the license files are copied into the `$HOME/.ds-5/licenses` folder.

## 3.5 Evaluating DS-5 Ultimate Edition

To evaluate DS-5, you can generate a license that allows you evaluate DS-5 Ultimate Edition for 30 days.

### Procedure

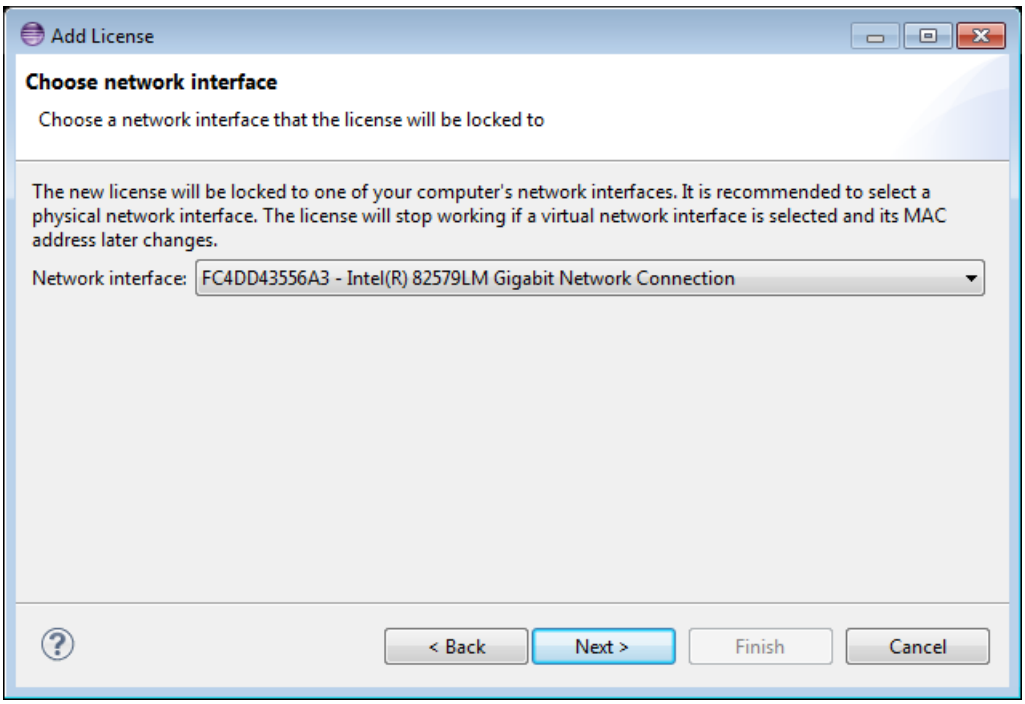
1. In the **Add License** dialog box, select **Generate a 30-day evaluation license for DS-5 Ultimate Edition**.

   ———— **Note** ————

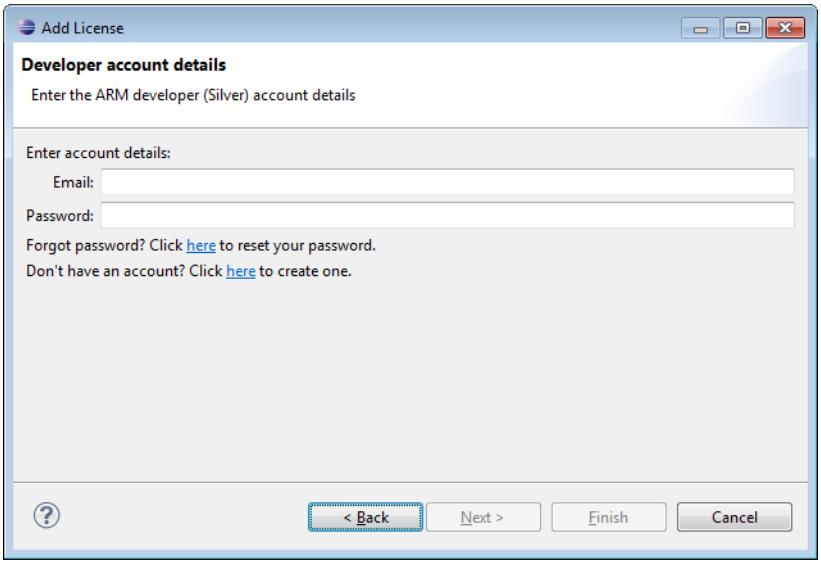
   Evaluation licenses are restricted to one 30-day evaluation license per machine. Contact your support team for extending your license.

   ————————————

2. Click **Next**.
3. In the **Choose network interface** page, select a **Network interface** from the drop-down list.




4. Click **Next**.
5. Enter your ARM developer account details in the ARM Self-Service Portal or if you do not have an account, you can create one.



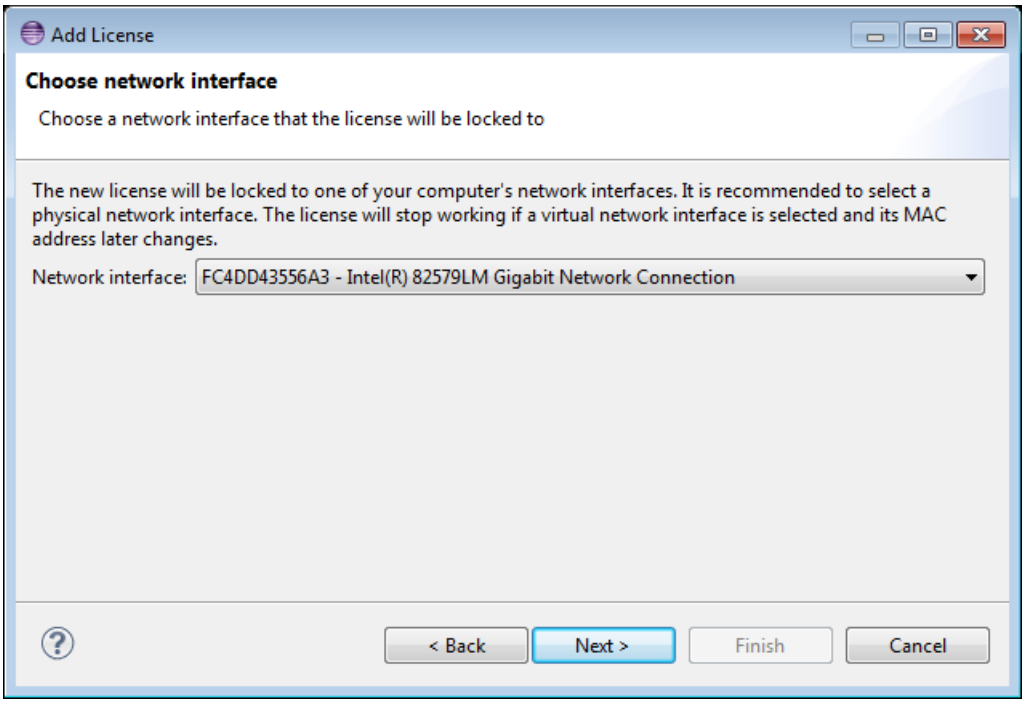

6. Click **Finish**.

## 3.6 Obtaining a license manually using the ARM website

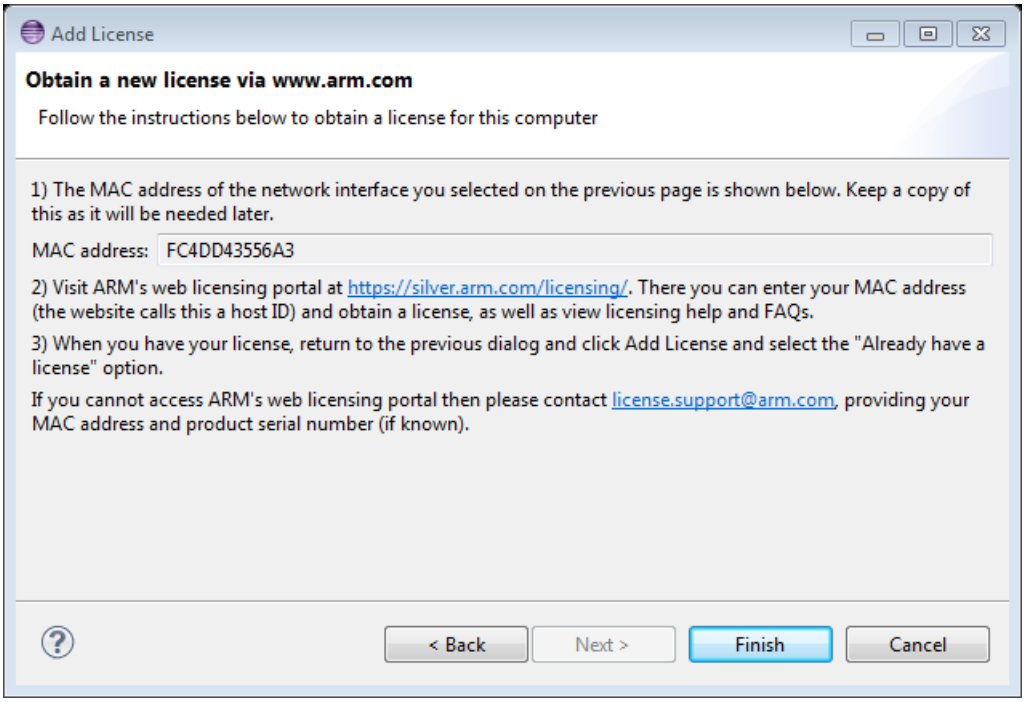
You can manually obtain a license from the ARM website.

### Procedure

1. In the **Add License** dialog box, select **Use a license file, license server, serial number, or activation code**. Then click **Next**.
2. In the **Obtain a new license** page, select **Manually obtain a license via www.arm.com website (advanced)**. Then click **Next**.
3. In the **Choose network interface** page, select a **Network interface** from the drop-down list.




4. Click **Next**
5. Follow the steps shown in the dialog box.



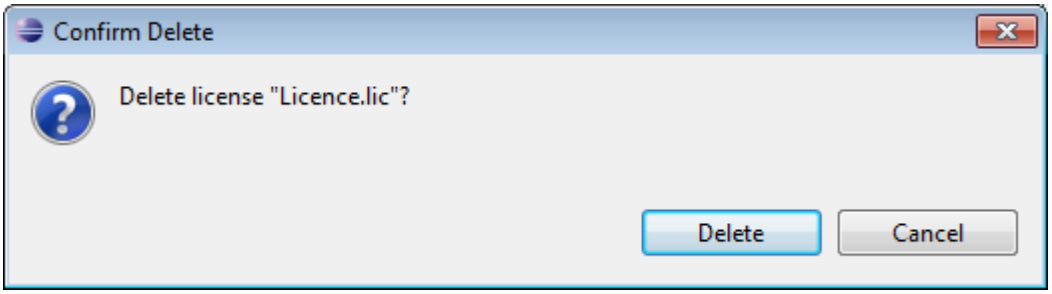

6. Click **Finish**.

## 3.7 Deleting a license

You can use the **Delete** option to delete a license.

### Procedure

1. To view the **ARM License Manager**, in Eclipse, select **Help** > **ARM License Manager...**.
2. In the **Configuration** tab of the ARM License Manager dialog box, select the license to be deleted.
3. Click **Delete License**.
4. In the **Confirm Delete** dialog box, click **Delete** to uninstall and remove the license file from the DS-5 license folder.

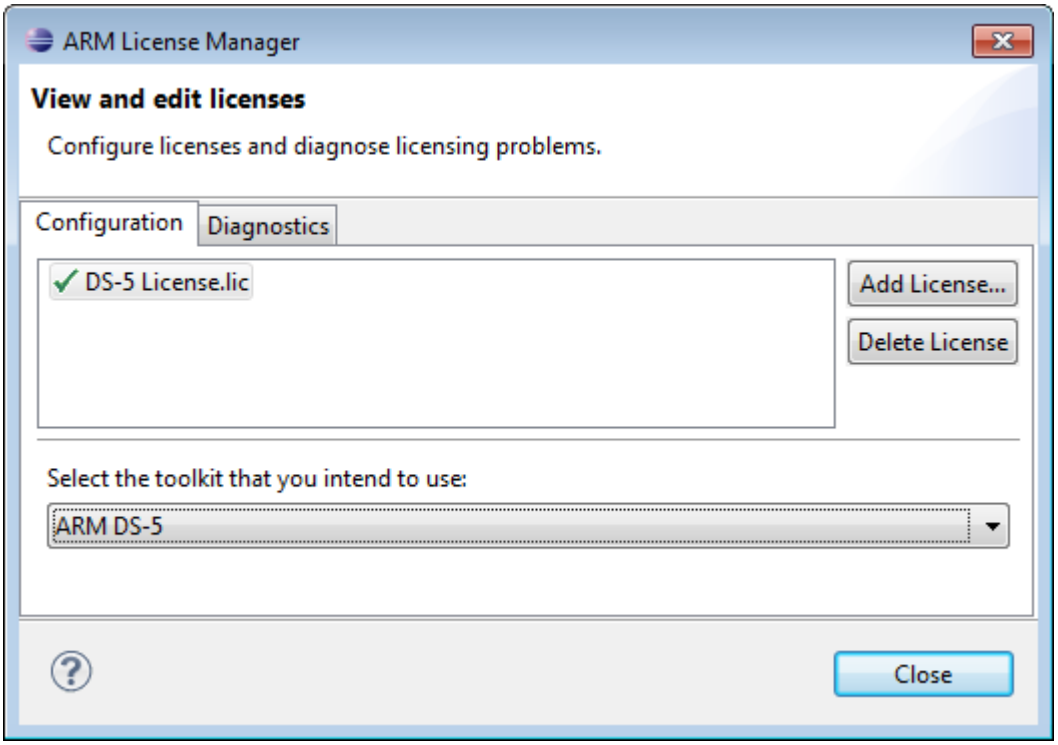


## 3.8 Changing the Toolkit

You can change the toolkit for DS-5 using the ARM License Manager.

### Procedure

1. Start Eclipse for DS-5.
2. Select **Help** > **ARM License Manager...**.
3. Click **Add License...** and follow the steps to add a license.
4. To change the toolkit, select it from the **Toolkit** drop-down menu.




**Note**

Some toolkit features are dependent on the installed license.

5. Click **Close** to close the dialog box.
6. Restart Eclipse.

### Related references

*3.1 Licensing and product updates* on page 3-30.

### Related information

*ARM DS-5 License Management Guide.*
*ARM Self-Service Portal.*

## 3.9 Viewing detailed license and system information

You can view system and DS-5 license information using the **Diagnostics** tab available in the ARM License Manager dialog box. Use this information to investigate licensing issues or to provide additional information to your support team.

### Procedure

1. To view the **ARM License Manager**, in Eclipse, select **Help** > **ARM License Manager...**.
2. Select the **Diagnostics** tab to view system and license information.
3. Click **Copy to Clipboard** to copy the information to the clipboard and send to your support team.
4. Click **Close** to close the dialog box.

# Chapter 4
# Working with ARM® DS-5

This chapter explains how to run and debug applications using ARM DS-5 tools. It also provides information about the examples and documentation provided with DS-5.

It contains the following sections:


- *4.1 Documentation provided with DS-5* on page 4-45.
- *4.2 Examples provided with DS-5* on page 4-46.
- *4.3 Importing the example projects into Eclipse* on page 4-48.
- *4.4 Creating a C or C++ project* on page 4-51.
- *4.5 Setting up the compilation tools for a specific build configuration* on page 4-53.
- *4.6 Building the project* on page 4-55.
- *4.7 Creating a new DS-5 debug configuration for an FVP connection* on page 4-56.
- *4.8 Running an application on an FVP with varying vector width* on page 4-60.
- *4.9 Building the gnometris project from Eclipse* on page 4-64.
- *4.10 Building the gnometris project from the command line* on page 4-65.
- *4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.
- *4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.
- *4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.
- *4.14 Launching gdbserver with an application* on page 4-74.
- *4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.
- *4.16 Debugging Gnometris* on page 4-78.
- *4.17 Debugging a loadable kernel module* on page 4-79.
- *4.18 Performance analysis of the threads application running on ARM® Linux* on page 4-83.
- *4.19 About registering a new compiler toolchain* on page 4-85.
- *4.20 Registering a compiler toolchain from the DS-5 command prompt* on page 4-86.
- *4.21 Configuring a compiler toolchain for the DS-5 command prompt* on page 4-90.

- *4.22 Registering a compiler toolchain from Eclipse* on page 4-91.

## 4.1 Documentation provided with DS-5

DS-5 includes example projects and documentation.

To access the documentation from within DS-5, from the main menu, select **Help** > **Help Contents** and navigate to **ARM DS-5 Documentation**.

If you prefer, the documentation is also *available on the web*.

Documentation on using the examples is available in *DS-5_install_directory*`\examples\docs`.

### Related references

*2.1 System requirements* on page 2-24.

*2.3 Installation directories* on page 2-28.

*3.1 Licensing and product updates* on page 3-30.

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*DS-5 documentation.*

## 4.2 Examples provided with DS-5

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

- Bare-metal software development examples for ARMv7 that illustrate:
  - Compilation with ARM Compiler 5.
  - Compilation with ARM Compiler 6.
  - Compilation with GCC bare-metal compiler.
  - ARMv7 bare-metal debug.

  The code is located in the archive file `<examples_directory>\Bare-metal_examples_ARMv7.zip`.
- Bare-metal software development examples for SVE using ARM Compiler 6.

  The code is located in the archive file `<examples_directory>\SVE_examples.zip`.
- Bare-metal software development examples for ARMv8 that illustrate:
  - Compilation with ARM Compiler 6.
  - Compilation with GCC bare-metal compiler.
  - ARMv8 bare-metal debug.

  The code is located in the archive file `<examples_directory>\Bare-metal_examples_ARMv8.zip`.

  **Note**

  ARMv8-A and ARMv8-R features are available only in the DS-5 Ultimate Edition. ARMv8-M features for microcontrollers are available in the DS-5 Professional and Ultimate Editions.
- 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 built with GCC Linux compiler that illustrate build, debug, and performance analysis of simple C/C++ console applications, shared libraries, and multi-threaded applications. These examples run on a *Fixed Virtual Platform* (FVP) that is preconfigured to boot ARM Linux. The files are located in the archive file, `examples_directory\Linux_examples.zip`.
- The *RTX Real-Time Operating System* (RTX-RTOS) source files and examples demonstrate the RTX-RTOS applications. The files are located in the archive file, `examples_directory\CMSIS_RTOS_RTX.zip`.
- Software examples for DS-5 Debugger's Debug and Trace Services Layer (DTSL). The examples are located in the archive file, `<examples_directory>\DTSL_examples.zip`.
- Jython examples for DS-5 Debugger. The examples are located in the archive file, `<examples_directory>\Jython_examples.zip`.
- The CoreSight™ Access Library is available as a github project at *https://github.com/ARM-software/CSAL*. A recent snapshot of the library from github is located in the archive file, `<examples_directory>\CoreSight_Access_Library.zip`.
- Optional packages with source files, libraries, and prebuilt images for running the examples can be downloaded from the DS-5 *downloads page*. You can also download the Linux distribution project with header files and libraries for the purpose of rebuilding the ARM Linux examples from the DS-5 downloads page.

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`.

### Related concepts

*1.4 About Fixed Virtual Platform (FVP)* on page 1-16.

### Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.

### Related references

*2.1 System requirements* on page 2-24.

*2.3 Installation directories* on page 2-28.

*3.1 Licensing and product updates* on page 3-30.

*4.1 Documentation provided with DS-5* on page 4-45.

### Related information

*Using the welcome screen.*

*ARM Development Studio 5 (DS-5).*

## 4.3 Importing the example projects into Eclipse

To use the example projects provided with DS-5, you must first import them.

### Procedure

1. Launch **Eclipse**:
   - On Windows, select **Start** > **All Programs** > **ARM DS-5** > **Eclipse for DS-5**.
   - On Linux, enter `eclipse` in the Unix bash shell.
2. ARM recommends that you create a workspace for example projects so that they remain separate from your own projects. To do this, you can either:
   - Create a workspace directory during the startup of Eclipse.
   - If Eclipse is already open, select **File** > **Switch Workspace** > **Other** from the main menu.
3. In the main menu, select **File** > **Import...**.
4. Expand the **DS-5** group.
5. Select **Examples and Programming Libraries** and click **Next**.

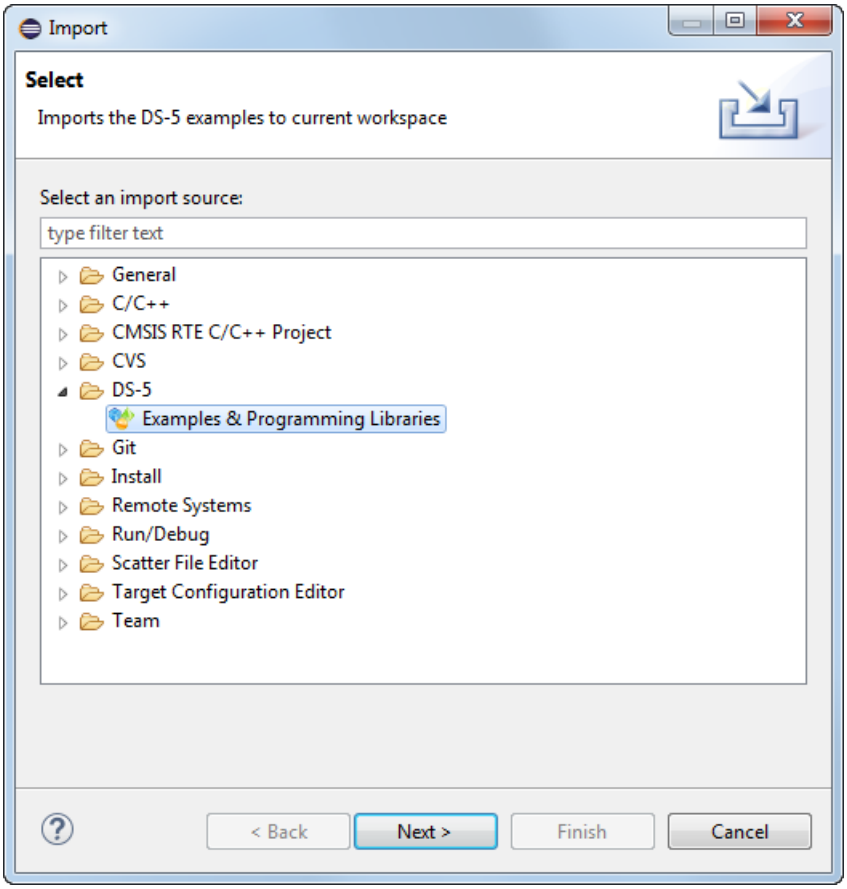



**Figure 4-1 Import DS-5 Examples and Programming Libraries**

6. Select the examples and programming libraries you want to import. If descriptions exist for examples, you can view it in the **Description** pane.

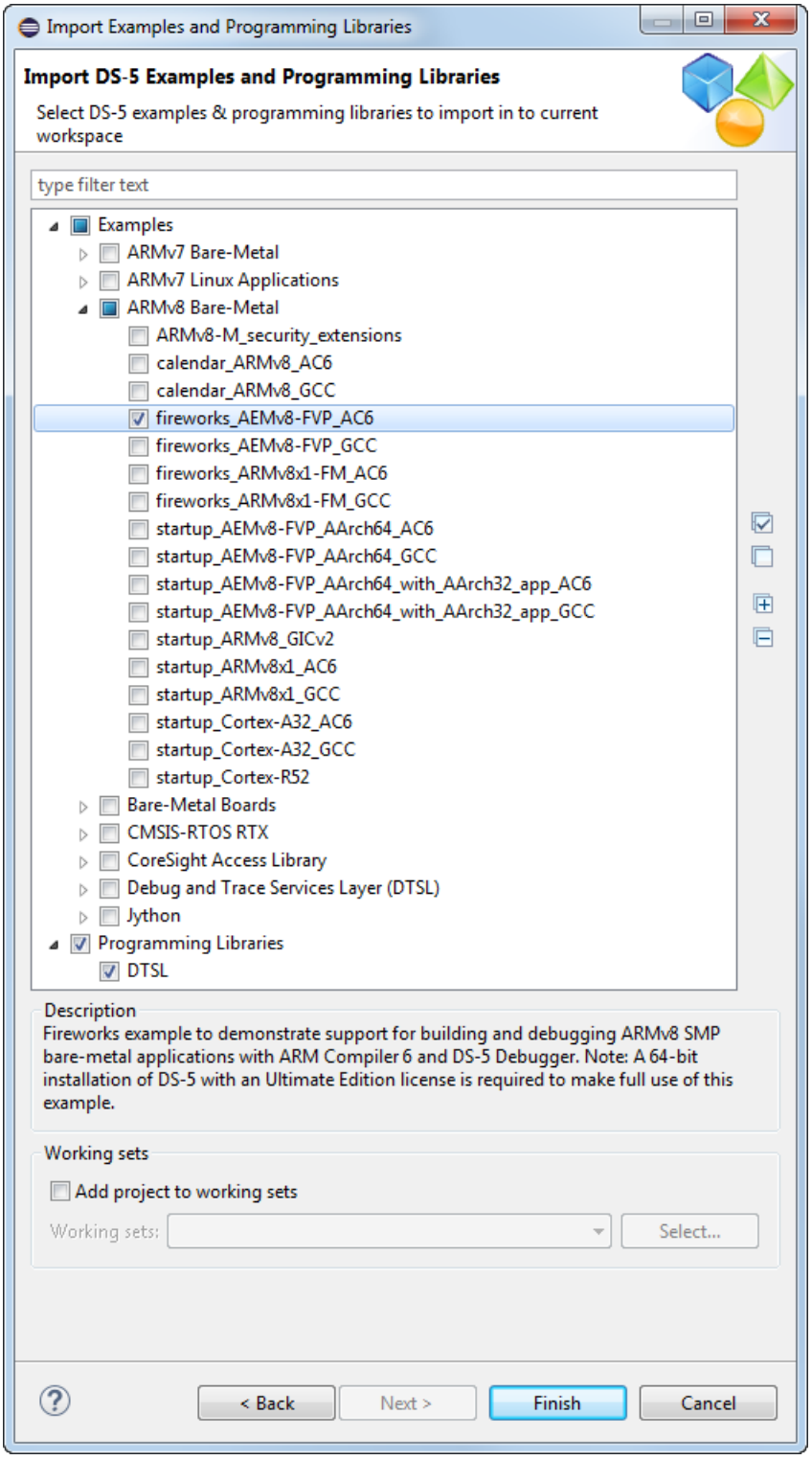



**Figure 4-2 Select DS-5 Examples and Programming Libraries**

7. If necessary, select **Add project to working sets** to add projects to a working set.
8. Click **Finish**.

You can browse the imported examples in the **Project Explorer**.

Each example contains a `readme.html` which explains how you can work with each example.

### Related tasks

*4.9 Building the gnometris project from Eclipse* on page 4-64.

*4.10 Building the gnometris project from the command line* on page 4-65.

*4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.

*4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.

*4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.

*4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.

*4.16 Debugging Gnometris* on page 4-78.

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*About working sets.*

*Creating a working set.*

*Changing the top level element when displaying working sets.*

*Deselecting a working set.*

## 4.4 Creating a C or C++ project

You can create projects in DS-5 using the **New Project** dialog box.

To create a new C or C++ Project:

### Procedure

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**, for example, `hello_world` .
6. Leave the **Use default location** option selected so that the project is created in the default folder shown. Alternatively, deselect this option and browse to your preferred project folder.
7. Select the type of project that you want to create, for example, **Executable** > **Empty Project**.

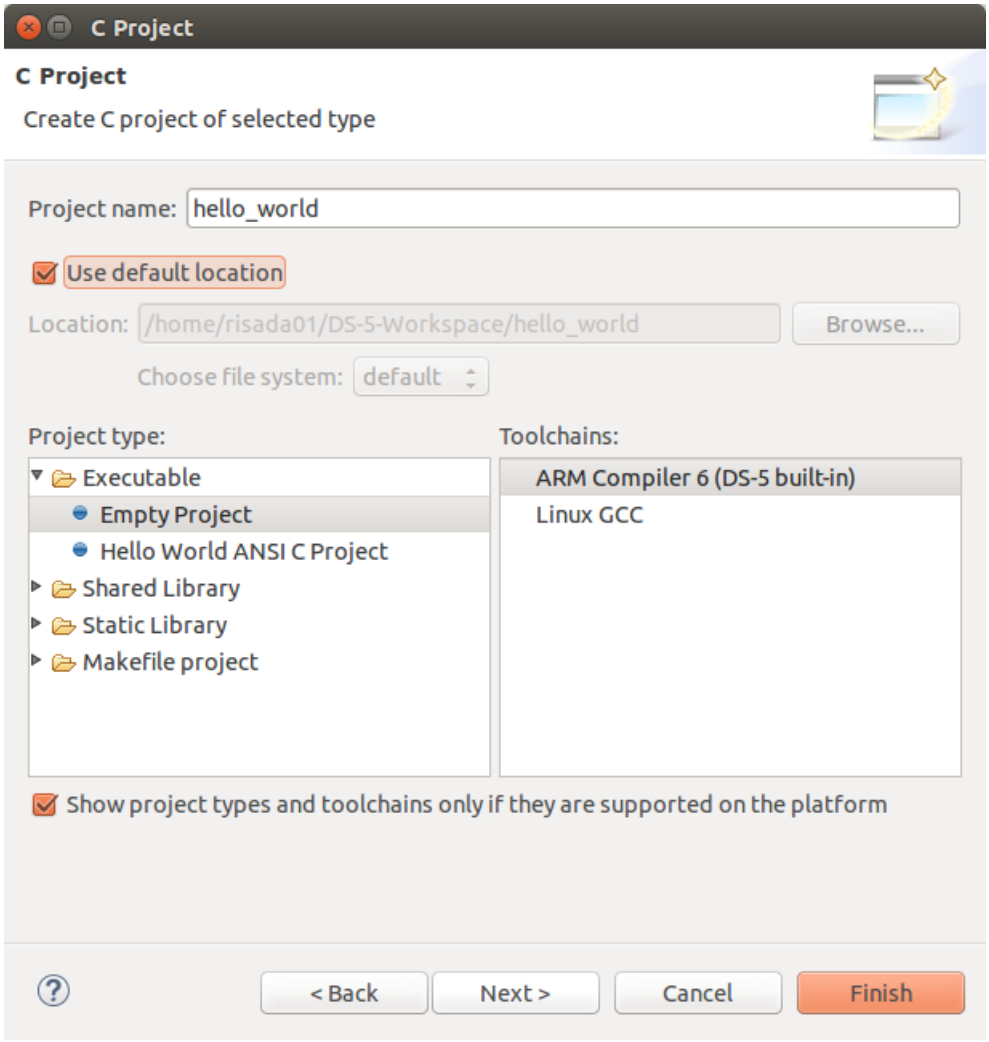



**Figure 4-3 Creating a new C project**

8. Select a **Toolchain**, for example, **ARM Compiler 6 (DS-5 built-in)**.
9. Click **Finish** to create your new project.

You can view the project in the **Project Explorer** view.

**Note**

**C project**
: This does not select a source language by default and leaves this decision to the compiler.

**C++ project**
: Selects C++ as the source language by default, regardless of file extension.

In both cases, the source language for the entire project, a source directory or individual source file can be configured in the build configuration settings.

## 4.5 Setting up the compilation tools for a specific build configuration

The C/C++ Build configuration panels enable you to set up the compilation tools for a specific build configuration, in your project. The settings in these panels determine how the compilation tools build an ARM executable image or library.

To access the build configuration panels:

### Procedure

1. Select the project in the **Project Explorer** view.
2. Select **Properties** from the **Project** menu.
3. Expand **C/C++ Build** in the **Properties** dialog box.
4. Select **Settings**.
5. The **Configuration** panel shows the active configuration. If required, click **Manage Configurations...** from the **Configuration** panel to create a new build configuration or change the currently selected configuration.
6. The compilation tools available for the current project, and their respective build configuration panels are displayed in the **Tool Settings** tab. Select this tab and configure the build as required.

   ——— **Note** ———

   Makefile projects do not use these configuration panels. The Makefile must contain all the required compilation tool settings.

   ———————————

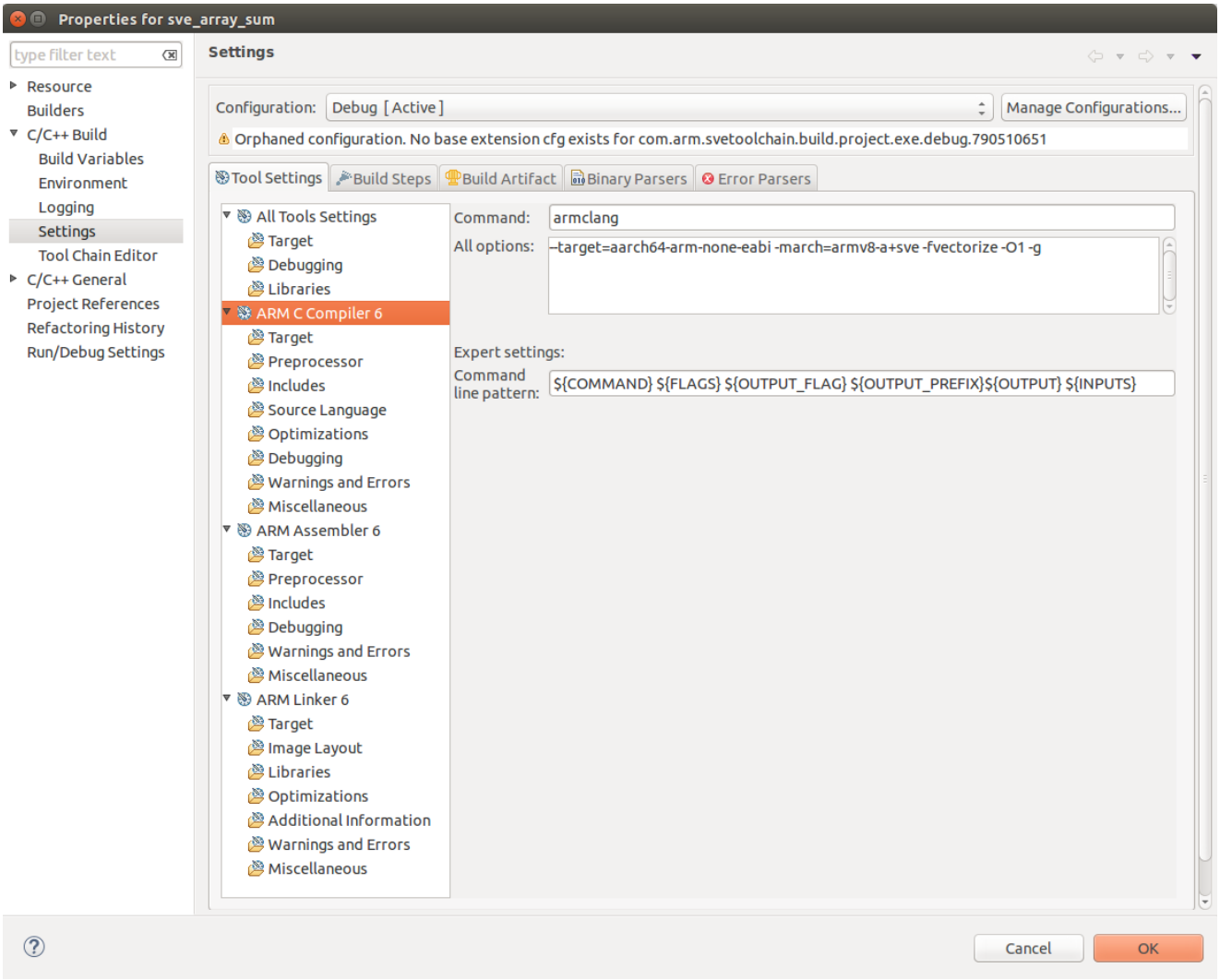



**Figure 4-4 Typical build settings dialog box for a C project**

7. To set optimization settings, select **ARM C Compiler 6** > **Optimizations** to display the optimization settings and change them.
8. To set a debug level, select **ARM C Compiler 6** > **Debugging** to display debug settings and change them.
9. Click **OK** to save the settings.

## 4.6 Building the project

You can add source files to your project and build the application using the compiler in DS-5. After a successful build, you can create a debug configuration and run the application on the target.

### Procedure

1. In the **Project Explorer** view, right-click your project, for example `hello_world`, and the select **File** > **New** > **Source File**.
2. In the **New Source File** dialog, enter the file name `hello.c`.
3. Click **Finish** to create the source file and open it in the code editing view. The source file is visible in the **Project Explorer** view, under the `hello_world` project.
4. Add the following code to the new source file, and save the file.

```
#include <stdio.h>
int main()
{
  printf("Hello world\n");
  return 0;
}
```

5. In the **Project Explorer** view, right-click on the `hello_world` project and select **Build Project**.

The `axf` file contains both the object code and debug symbols that enable the debugger to perform source-level debugging.

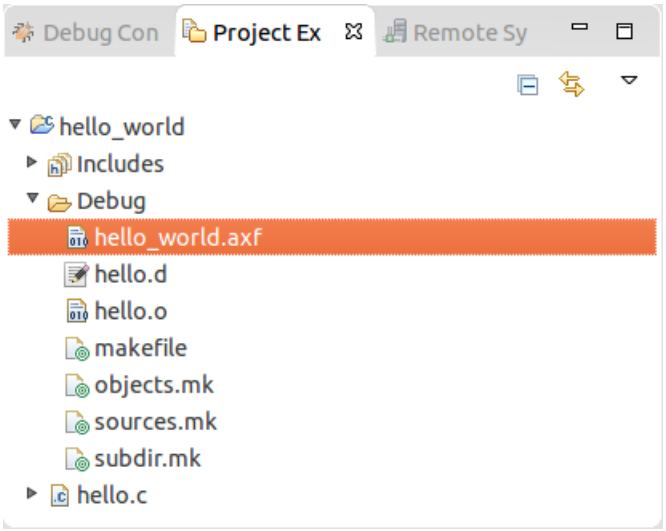



**Figure 4-5 Compiled axf file**

## 4.7 Creating a new DS-5 debug configuration for an FVP connection

This procedure describes how to create a new debug configuration in DS-5. You can use the debug configuration to connect and run your application on an FVP model or on a hardware target.

### Procedure

1. From the DS-5 main menu, select **Run** > **Debug Configurations**.
2. In the left-hand panel of the **Debug Configurations** dialog, select .

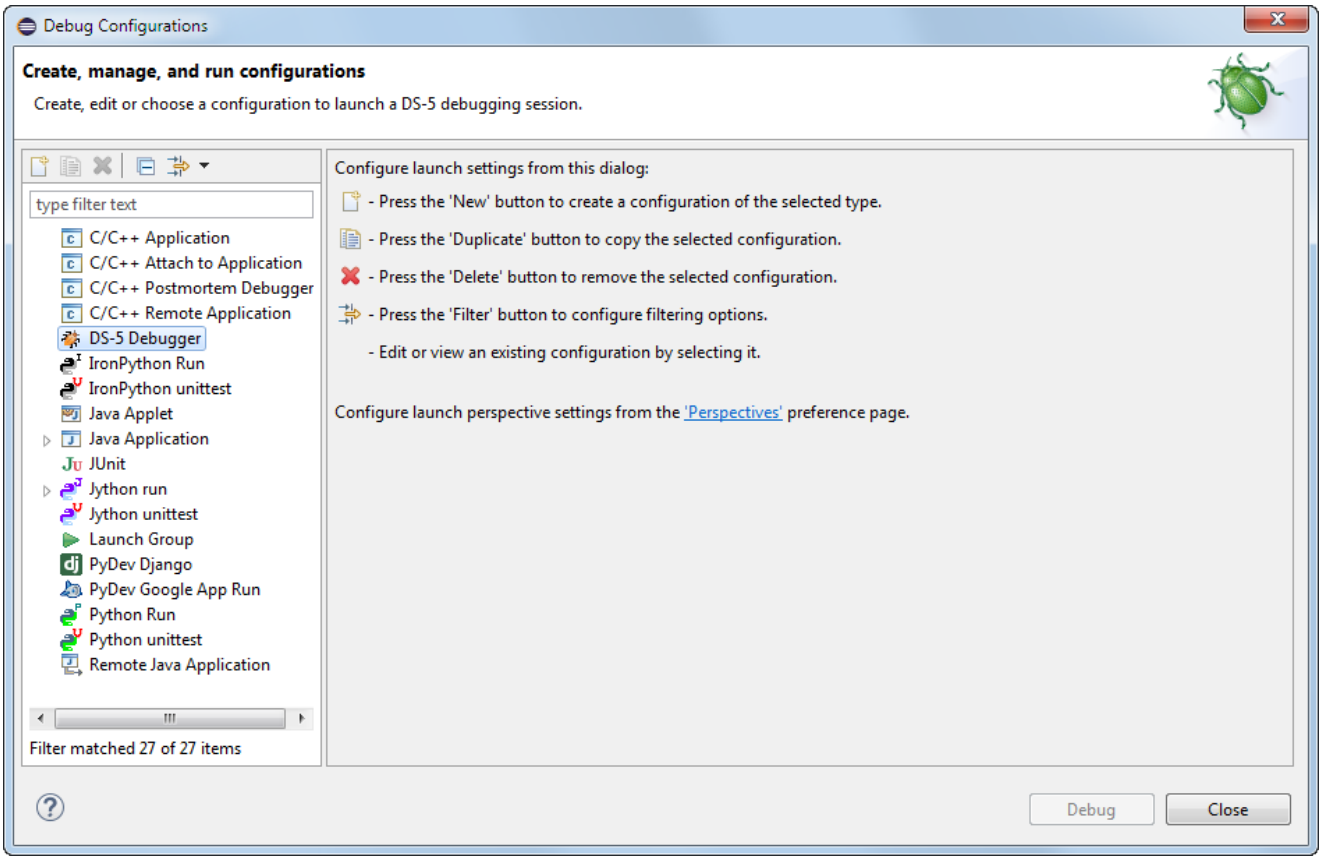



**Figure 4-6 Create a new debug configuration**

3. Click the **New** button to create a new DS-5 Debugger launch configuration. This displays the various tabs required to specify settings for loading your application on the target.
4. Give a name to the debug configuration. For example, `hello_world`.
5. In the **Connection** tab, select **ARM FVP (Installed with DS-5)** > **ARMv8-Ax1 Foundation Platform** > **Bare-Metal Debug** > **Debug ARMv8-A**.

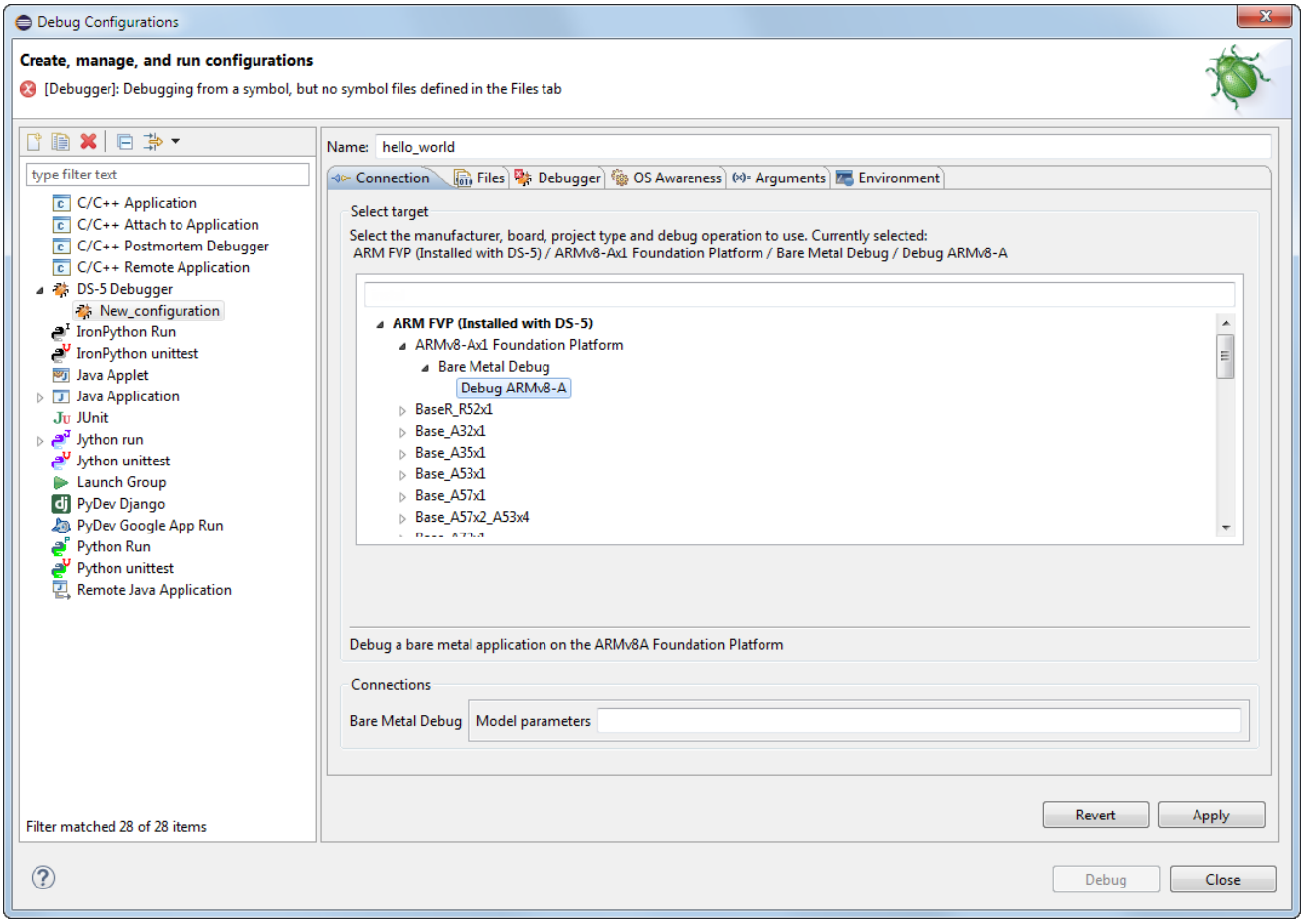



**Figure 4-7 Select an FVP model to connect to**

**Tip**

You can enter a filter name, for example `sve` in the **Filter Platforms** field to help find the target you want to connect to.

**Note**

If you are connecting to a SVE enabled FVP model, ensure that the **Model parameters** field at the bottom of the debug configurations dialog is set to `--plugin DS5://sw/models/bin/ScalableVectorExtension.so -C bp.secure_memory=false`.

6. Select the **Files** tab, and under for the **Application on host to download** field, click **Workspace**. This displays all the projects in your current workspace.
7. Select your project, which you have built earlier, and within it, select the image you want to load, for example `hello_world.axf`. Then click **OK**.

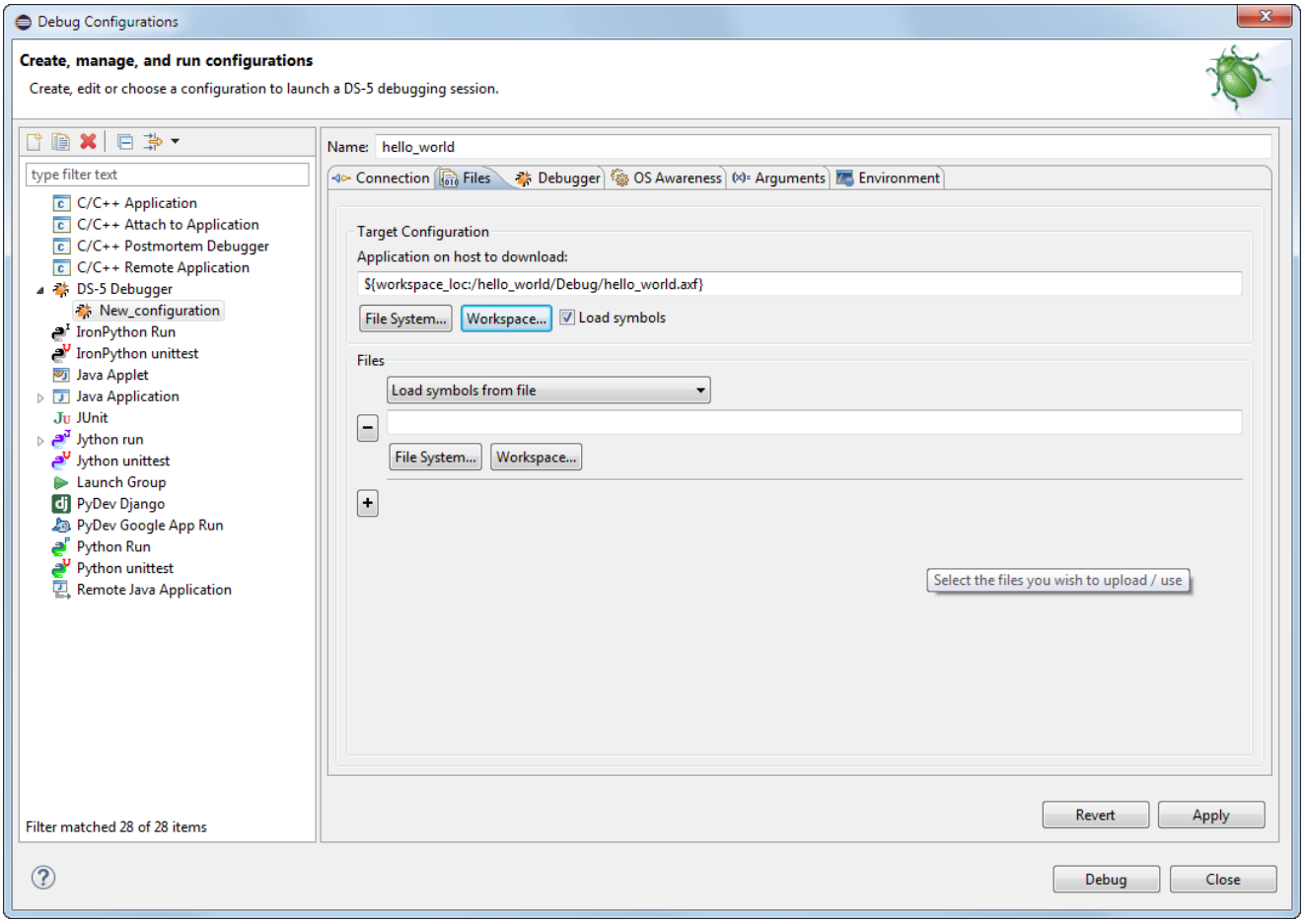



**Figure 4-8 Specify the application to load**

8. Select the **Debugger** tab, and then select the **Debug from symbol** option. Set the symbol to **main**.

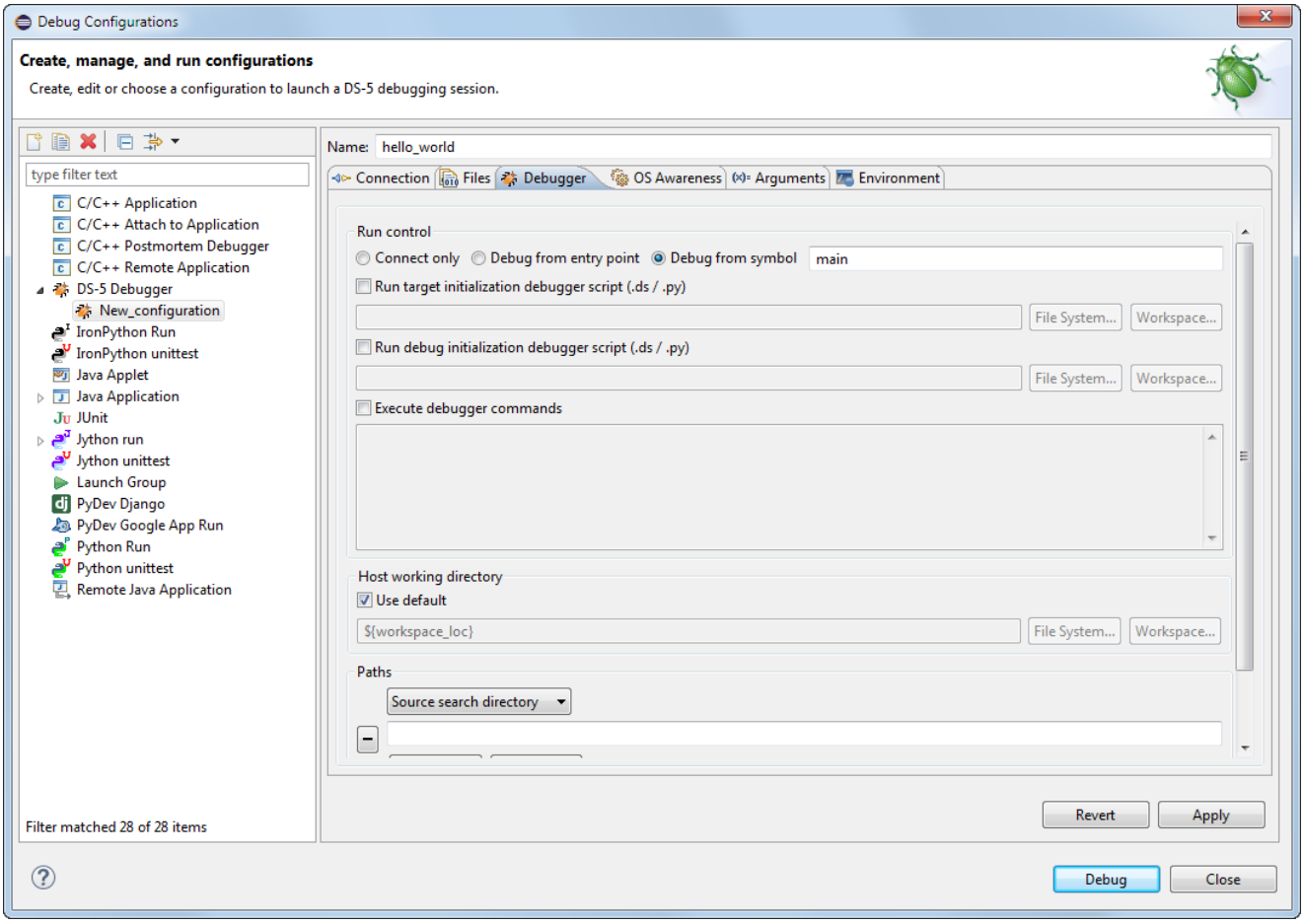



**Figure 4-9 Specify the debug symbol**

9. Click **Debug** to connect and load the application on the FVP model. This launches the FVP model and also loads the debug information into the debugger. DS-5 also displays the connection status in the **Debug Control** view.
10. If a **Confirm Perspective Switch** dialog appears, click **Yes**.
11. The application stops at the `main()` function and is ready to run or debug.

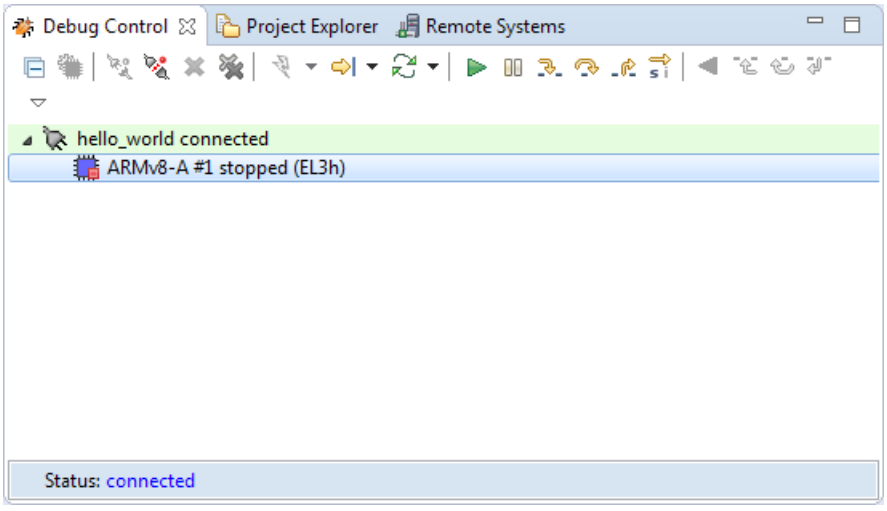



**Figure 4-10 Debug Control view**

12. Click the **Continue** button to continue running the application. You can view the application output in the **Target Console** view.

## 4.8 Running an application on an FVP with varying vector width

This describes running a simple program that has been compiled using ARM Compiler 6. It shows how you can increase the SVE vector width on the FVP model to reduce the total number of executed instructions.

This example uses a project, `sve_array_sum`, that is provided with DS-5, which you can import. The application contains two vectorizable loops. The first loop fills the array, `values`, with floating-point values. The second loop adds all these values together.

### Procedure

1. In the **Project Explorer** view, select the project `sve_array_sum`. If the project does not contain the compiled `.axf` file, then right-click the project and build it using ARM Compiler 6.
2. DS-5 provides an FVP debug configuration for this project, `sve_array_sum_FVP`. To open the debug configuration, right-click the project and select **Debug As** > **Debug Configurations** and select `sve_array_sum_FVP` from the list of **DS-5 Debugger** configurations.
3. In the **Model parameters** field, specify these three options:
   - *`-C bp.secure_memory=false`*
   - `-C SVE.ScalableVectorExtension.veclen=2`
   - `--stat`

   The `veclen` option defines the SVE vector width, in units of 64-bit (8-byte) blocks. The maximum value of `veclen` is 32, which corresponds to the architectural maximum SVE vector width of 2048-bit (256-byte) blocks. The SVE architecture only supports vector lengths in 128-bit (16 byte) blocks, so all values of `veclen` must be even. For example, a `veclen` value of 8 signifies 512-bit vector width SVE registers.

   The `--stat` option prints statistics about the application when the FVP model exits.

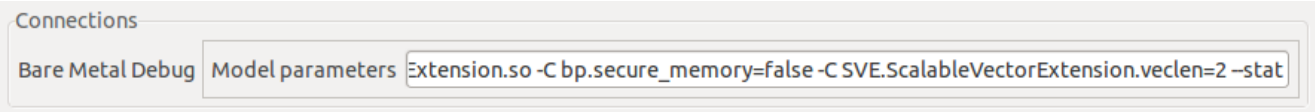



**Figure 4-11 Model parameters for vector length 128 bits (veclen=2)**

4. Click **Apply** and then click **Debug** to load the application on the FVP model.
5. To see the SVE registers, open the **Registers** view from the main menu **Windows** > **Show View**. If the SVE registers are not visible, in **Register set**, select **All registers** and then expand **AARCH64** > **SVE** > **Data**.

   When `veclen` is set to 2, the SVE Z registers are 128-bit registers.

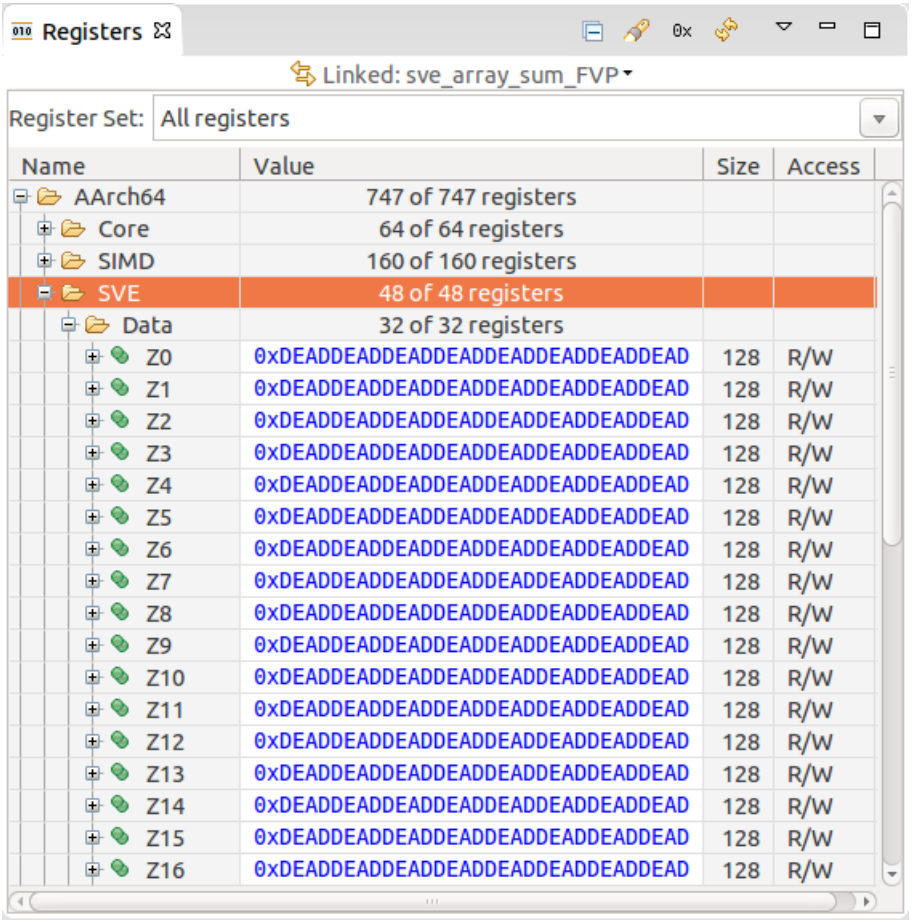



**Figure 4-12 SVE registers with vector length 128 bits (veclen=2)**

6. Using the **Debug Control** view, you can step through the application or run it till a breakpoint is hit. You can open the **Debug Control** view using **Window** > **Show View**.
7. Using the **Debug Control** view, run the application to completion. Then right-click the connection and select **Disconnect from Target** to disconnect from the model.
8. Open the **Target Console** view from the main menu **Window** > **Show View**.

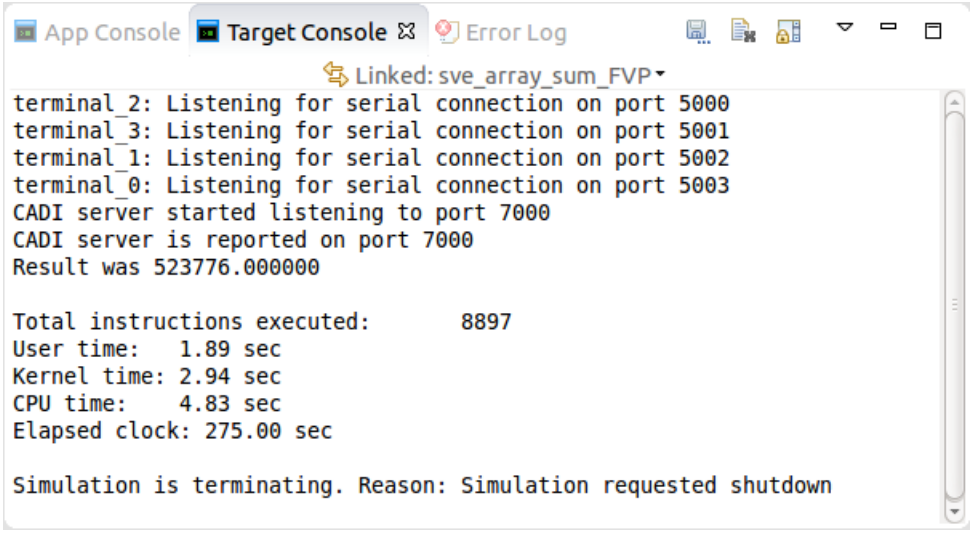



**Figure 4-13 Target console with vector length 128 bits (veclen=2)**

**Note**

The statistics output shows `Total instructions executed: 8897`. This number might be different depending on the other compiler settings.

9. To change the vector width for the FVP model, open the debug configurations for the project. In the **Model parameters**, change the `veclen` parameter to 32. Then click **Debug**.

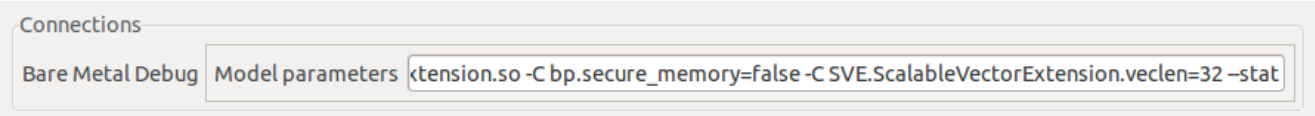



**Figure 4-14 Model parameters for vector length 2048 bits (veclen=32)**

10. Go to the **Registers** view and note that the SVE Z registers are 2048-bit registers when `veclen` is 32.

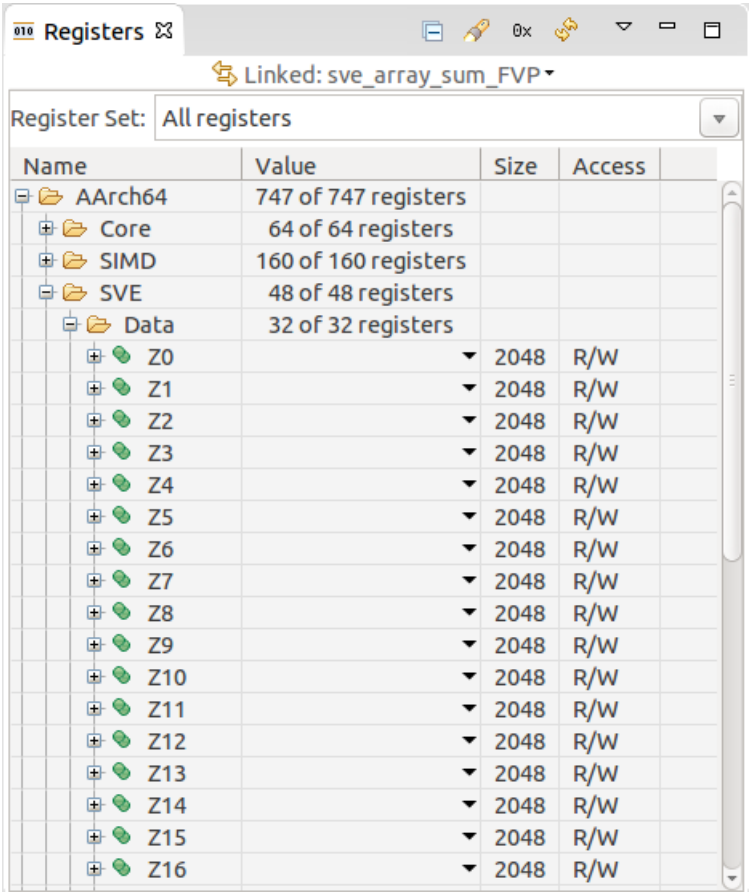



**Figure 4-15 SVE registers with vector length 2048 bits (veclen=32)**

11. Run the application to completion and then disconnect from the target. In the **Target Console** view, see that the total instructions executed is lower. The wider the SVE vector, the fewer instructions are needed to process the array.

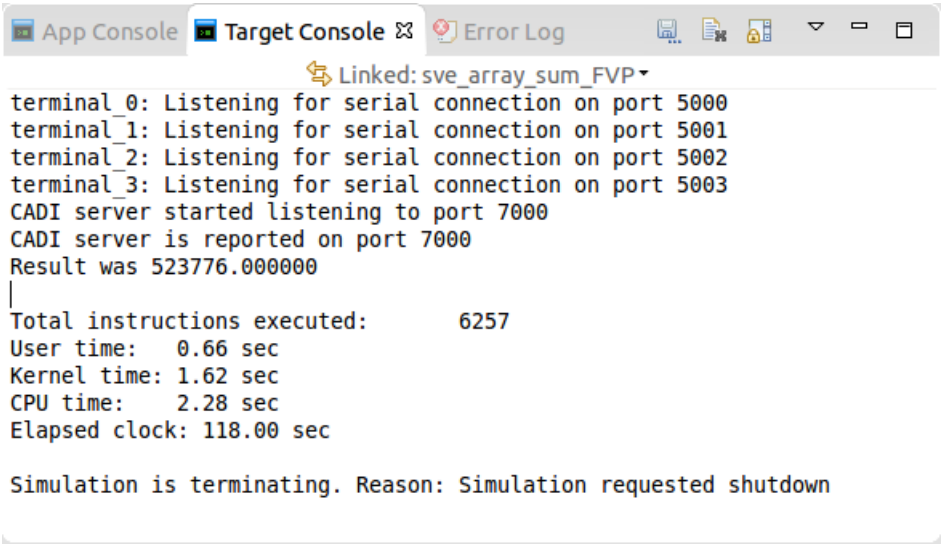



**Figure 4-16 Target console with vector length 2048 bits (veclen=32)**

## 4.9 Building the gnometris project from Eclipse

`gnometris` is an ARM Linux application that you can run and debug on your target. The supplied project contains prebuilt image binaries. Use these instructions to rebuild the project.

### Procedure

1. *Import* on page 4-48 the ARMv7 Linux application examples into your workspace. These include the `gnometris` example.
2. *Download* and copy the DS-5 Linux example distribution into your workspace.
3. Select the `gnometris` project in the **Project Explorer** view.
4. Select **Build Project** from the **Project** menu.

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

When you build the `gnometris` project, it produces the following applications:

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

### Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.

*4.10 Building the gnometris project from the command line* on page 4-65.

*4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.

*4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.

*4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.

*4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.

*4.16 Debugging Gnometris* on page 4-78.

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*Working with projects.*

## 4.10 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 contains prebuilt image binaries. Use these instructions to rebuild the project from the command line.

### Procedure

1. *Download* the DS-5 Linux distribution examples (`Linux_distribution_example.zip`) and extract it into the working directory.
2. Extract the contents of the `Linux_examples.zip` archive file located in *`DS-5_install_directory`*`/examples/` into the working directory. This zip file includes the `gnometris` example project source.
3. Open the **DS-5 Command Prompt** command-line console or a Unix bash shell.
4. Navigate to `../<working-directory>/gnometris`.
5. At the prompt, enter `make`.

   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.

### Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.

*4.9 Building the gnometris project from Eclipse* on page 4-64.

*4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.

*4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.

*4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.

*4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.

*4.16 Debugging Gnometris* on page 4-78.

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

## 4.11 Loading the Gnometris application on a *Fixed Virtual Platform* (FVP)

You can load the Gnometris application on to an FVP that is running ARM Linux.

An FVP enables you to run and debug applications on your host workstation without using any hardware targets.

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

### Procedure

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

### Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.
*4.9 Building the gnometris project from Eclipse* on page 4-64.
*4.10 Building the gnometris project from the command line* on page 4-65.
*4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.
*4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.
*4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.
*4.16 Debugging Gnometris* on page 4-78.

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*Configuring a connection to an FVP.*
*Debug Configurations - Connection tab.*
*Debug Configurations - Files tab.*
*Debug Configurations - Debugger tab.*
*Debug Configurations - Environment tab.*

## 4.12 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.

### Procedure

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:
   - 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.
   - Use an external file transfer utility such as `PuTTY`.

### Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.
*4.9 Building the gnometris project from Eclipse* on page 4-64.
*4.10 Building the gnometris project from the command line* on page 4-65.
*4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.
*4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.
*4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.
*4.16 Debugging Gnometris* on page 4-78.

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*Debug Configurations - Connection tab.*
*Debug Configurations - Files tab.*
*Debug Configurations - Debugger tab.*
*Debug Configurations - Environment tab.*
*Target management terminal for serial and SSH connections.*
*Remote Systems view.*

## 4.13 Configuring 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.

### Procedure

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

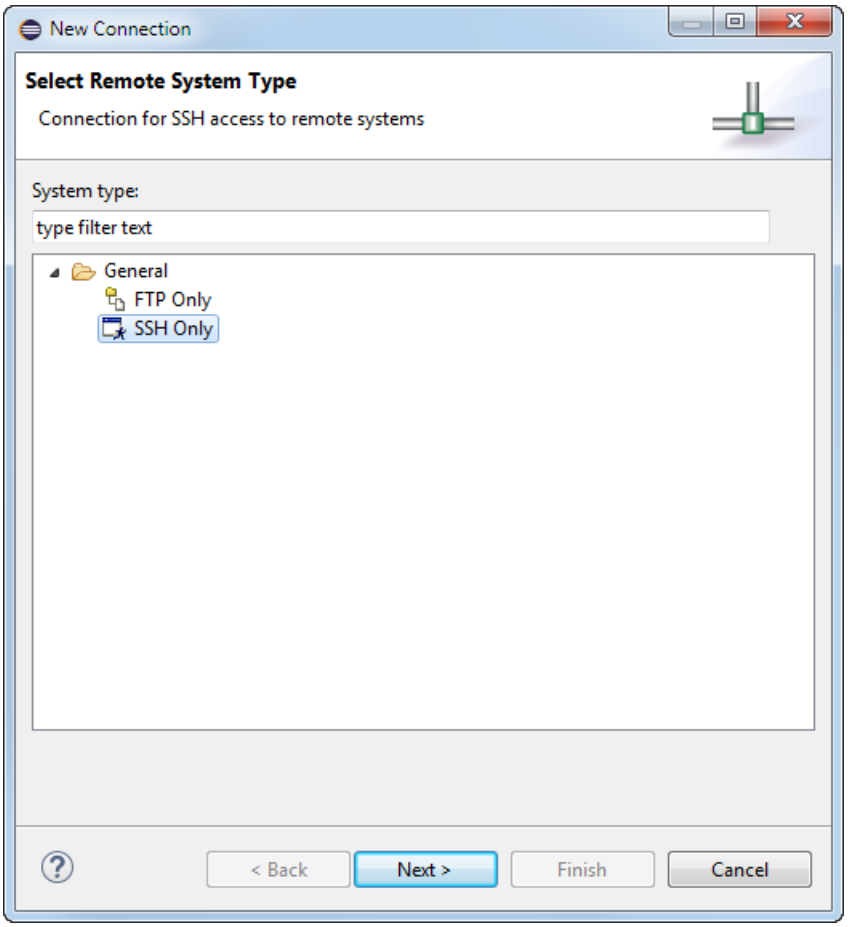



**Figure 4-17 Selecting a connection type**

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

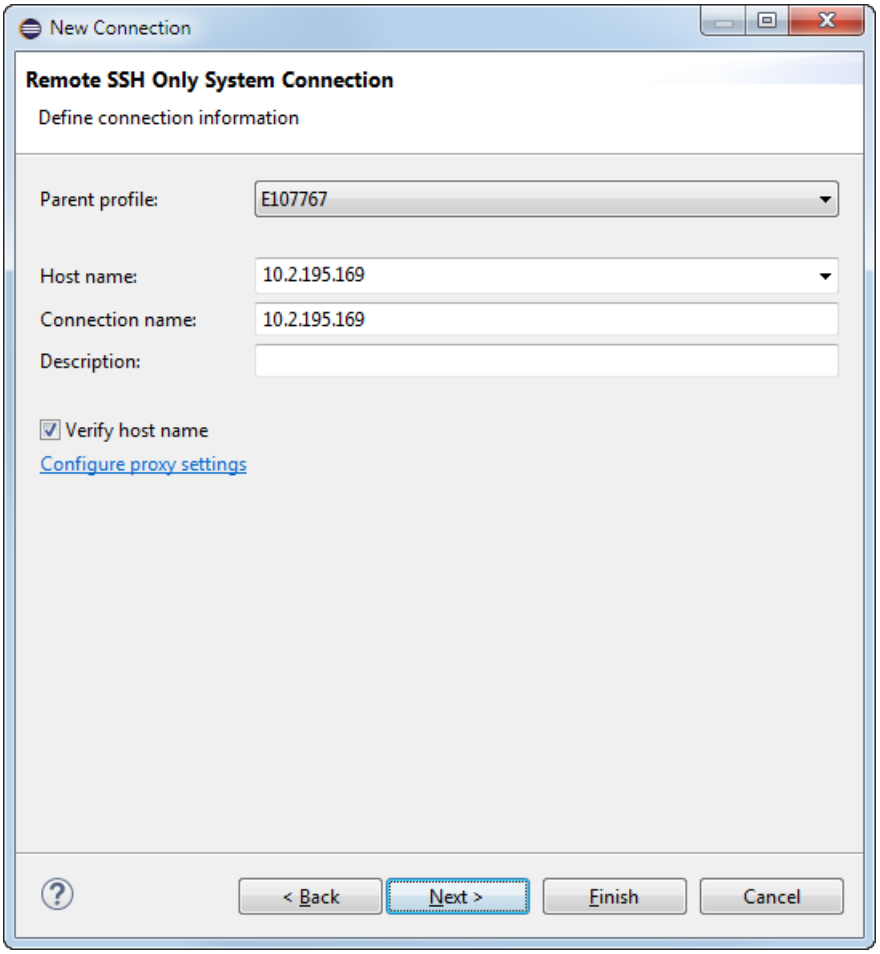



**Figure 4-18 Enter connection information**

5. Click **Next**.
6. Verify if the **Sftp Files Configuration** and **Available Services** are what you require.

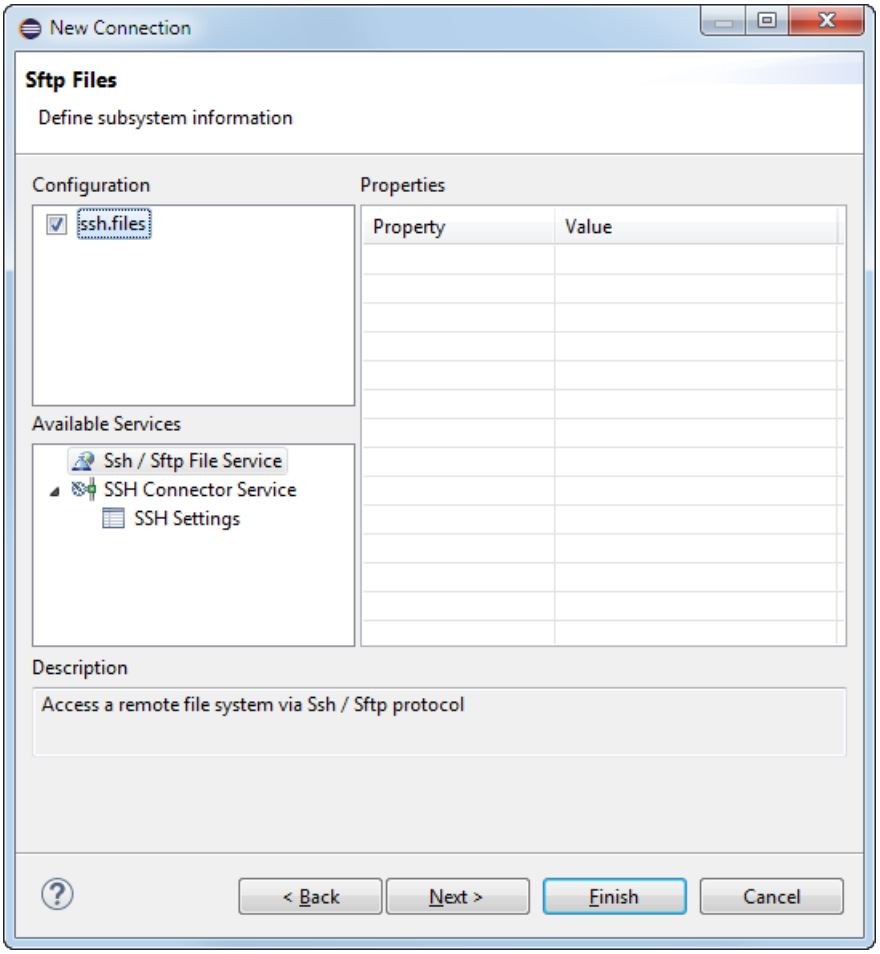



**Figure 4-19 Sftp Files options**

7. Click **Next**.
8. Verify if the **Ssh Shells Configuration** and **Available Services** are what you require.

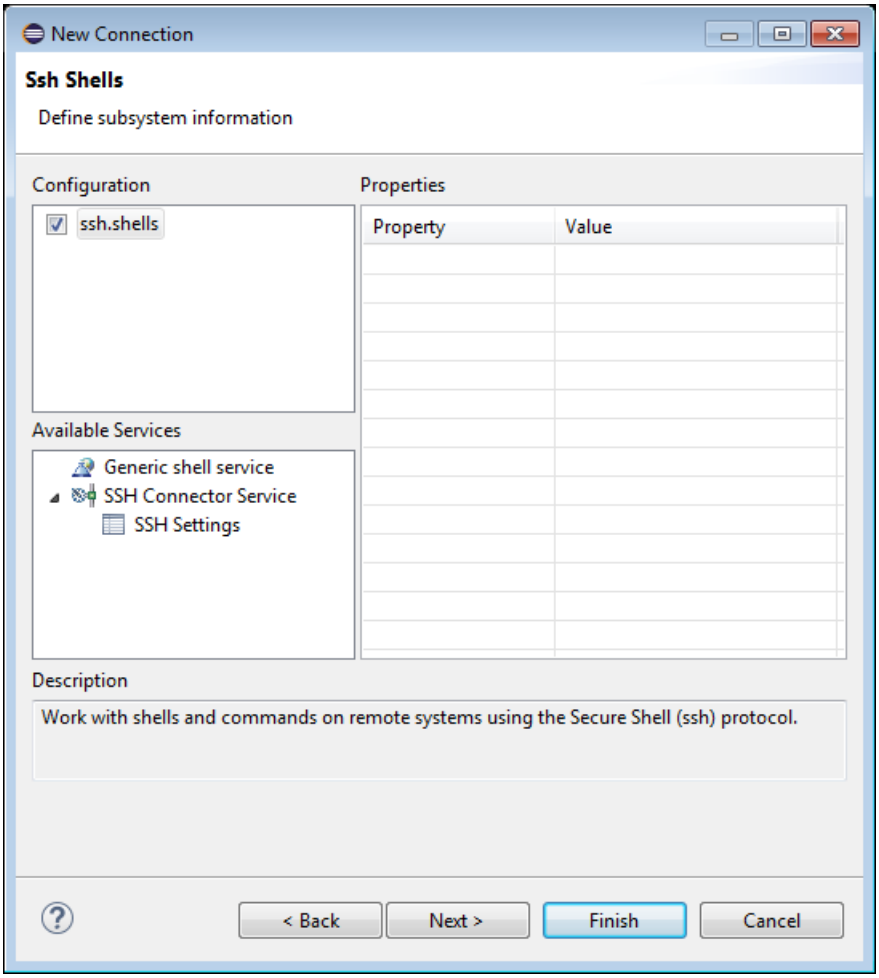



**Figure 4-20 Defining the shell services**

9. Click **Next**.
10. Verify if the **Ssh Terminals Configuration** and **Available Services** are what you require.

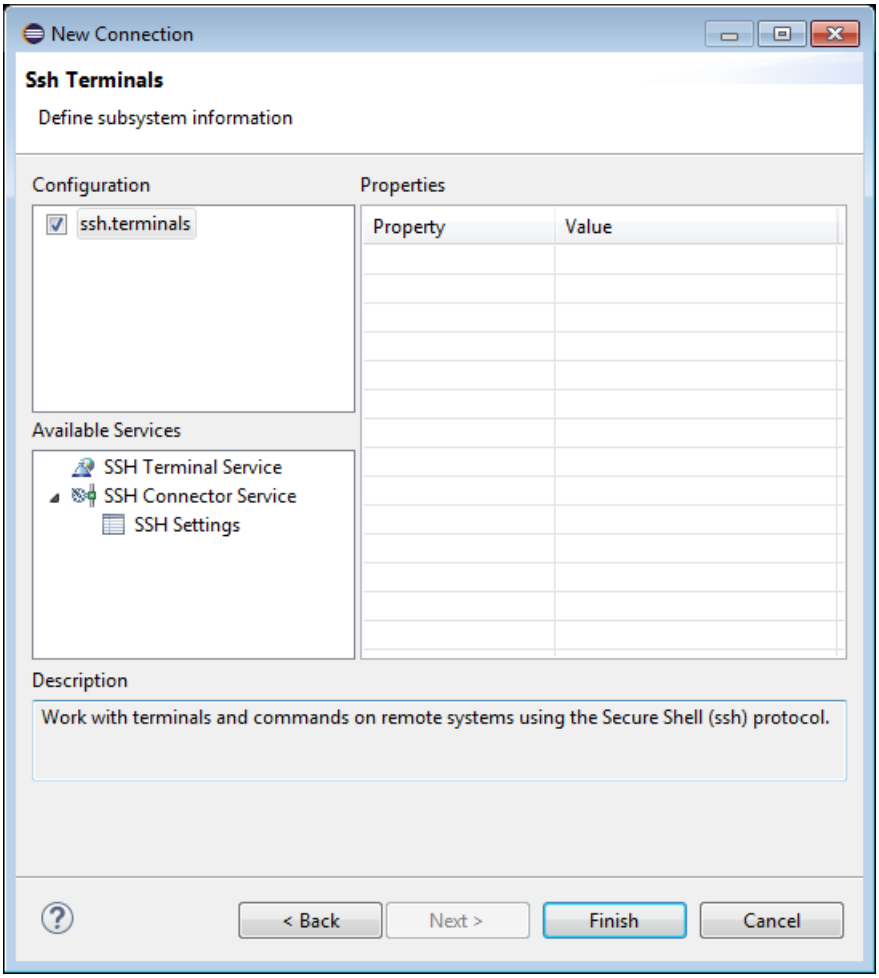



**Figure 4-21 Defining the terminal services**

11. Click **Finish**.
12. In the **Remote Systems** view:
    a. Right-click on the target and select **Connect** from the context menu.
    b. In the **Enter Password** dialog box, enter a **UserID** and **Password** if required.
    c. Click **OK** to close the dialog box.

Your SSH connection is now set up. You can copy any required files 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.

### Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.
*4.9 Building the gnometris project from Eclipse* on page 4-64.
*4.10 Building the gnometris project from the command line* on page 4-65.
*4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.
*4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.
*4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.
*4.16 Debugging Gnometris* on page 4-78.

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*Debug Configurations - Connection tab.*
*Debug Configurations - Files tab.*
*Debug Configurations - Debugger tab.*
*Debug Configurations - Environment tab.*
*Target management terminal for serial and SSH connections.*
*Remote Systems view.*

## 4.14 Launching gdbserver with an application

To launch `gdbserver` with the application:

### Procedure

1. Open a terminal shell that is connected to the target.
2. In the Remote Systems view, right-click on **Ssh Terminals**.
3. Select **Launch Terminal** to open a terminal shell.
4. In the terminal shell, navigate to the directory where you copied the application, then execute the required commands.

   For example, to launch Gnometris:

   ```
   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 connected to it then you do not need to use the `export DISPLAY` command.

## 4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target

Describes how to connect to the Gnometris application that is already running on a ARM Linux target.

### Prerequisites

- *gdbserver* and the Gnometris application running on the target and awaiting a connection on the appropriate port.
- The Gnometris application files available in your host workspace.

### Procedure

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 and:
   a. In the **Select target** panel, browse and select **Linux Application Debug** > **Connections via gdbserver** > **Connect to already running application**.
   b. In the **Connections** panel, enter the TCP **Address** and **Port** details of the *gdbserver* running on the target system.

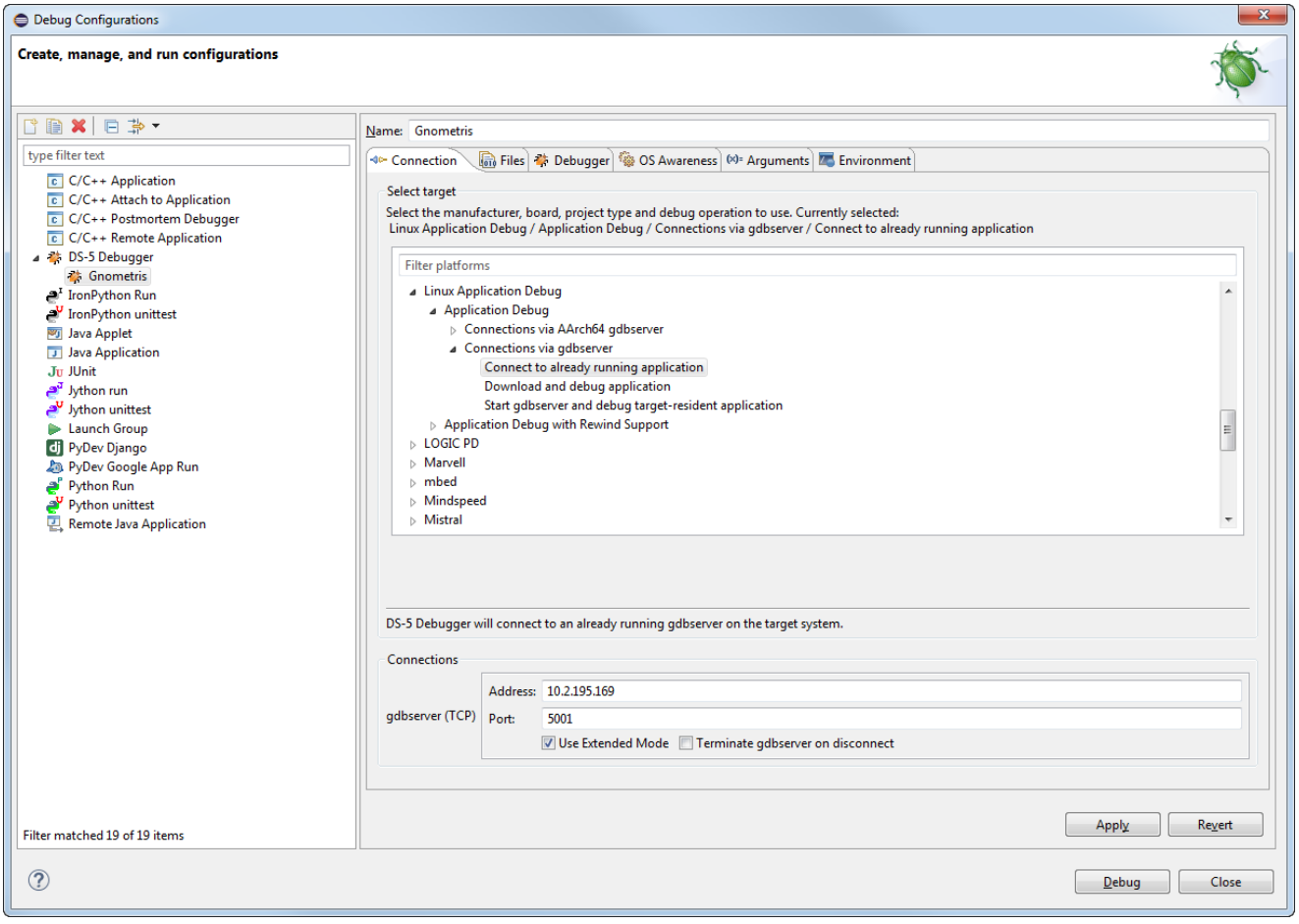



**Figure 4-22 Typical connection configuration for Linux application debug**

5. Click on the **Files** tab and:

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`.

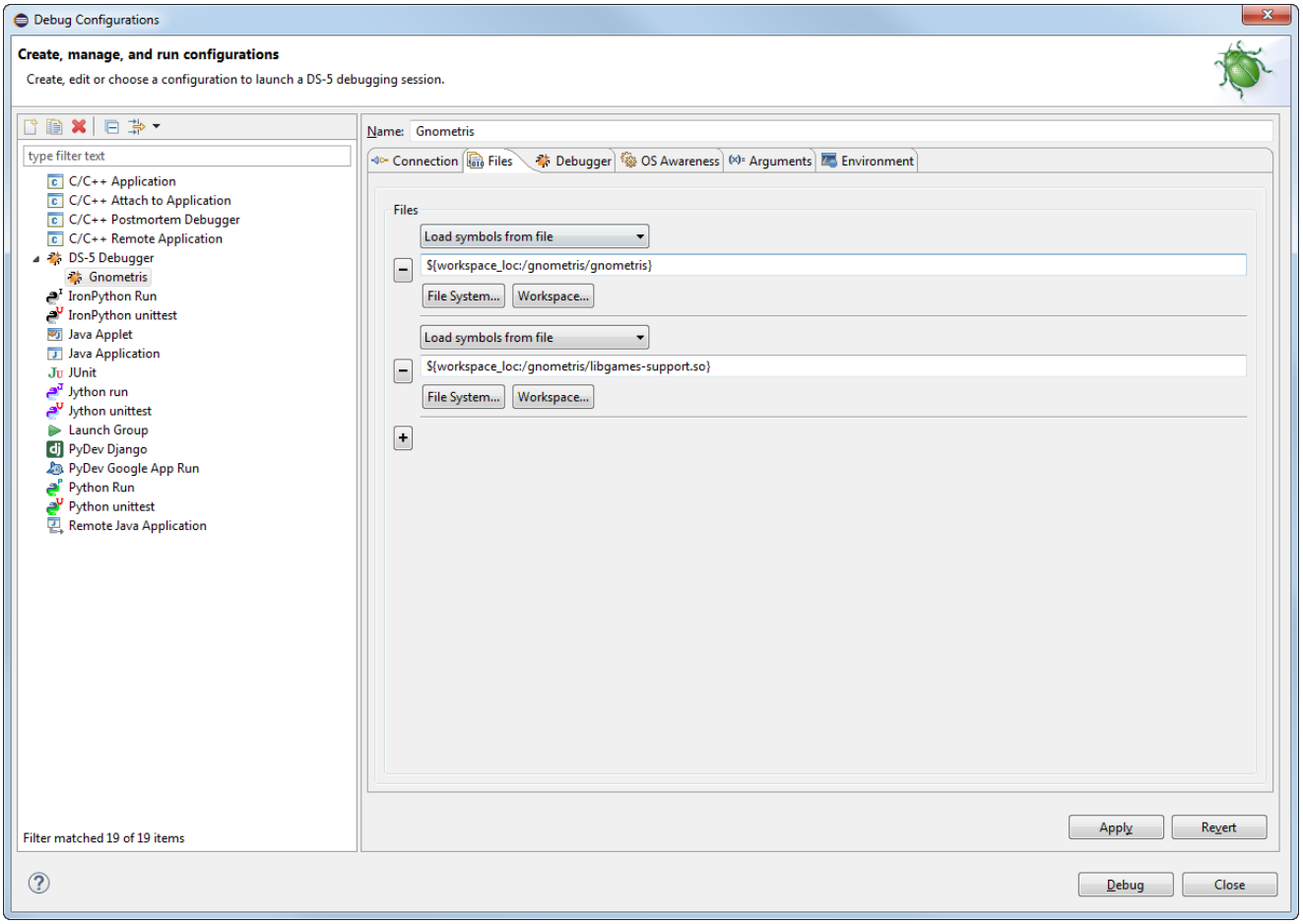



**Figure 4-23 Typical file selection for Linux application debug**

6. Click on the **Debugger** tab, and:
   a. In the **Run control** panel, select **Debug from symbol**.
   b. Enter **main** in the field provided.
7. In the **Host working directory** panel, select **Use default**.

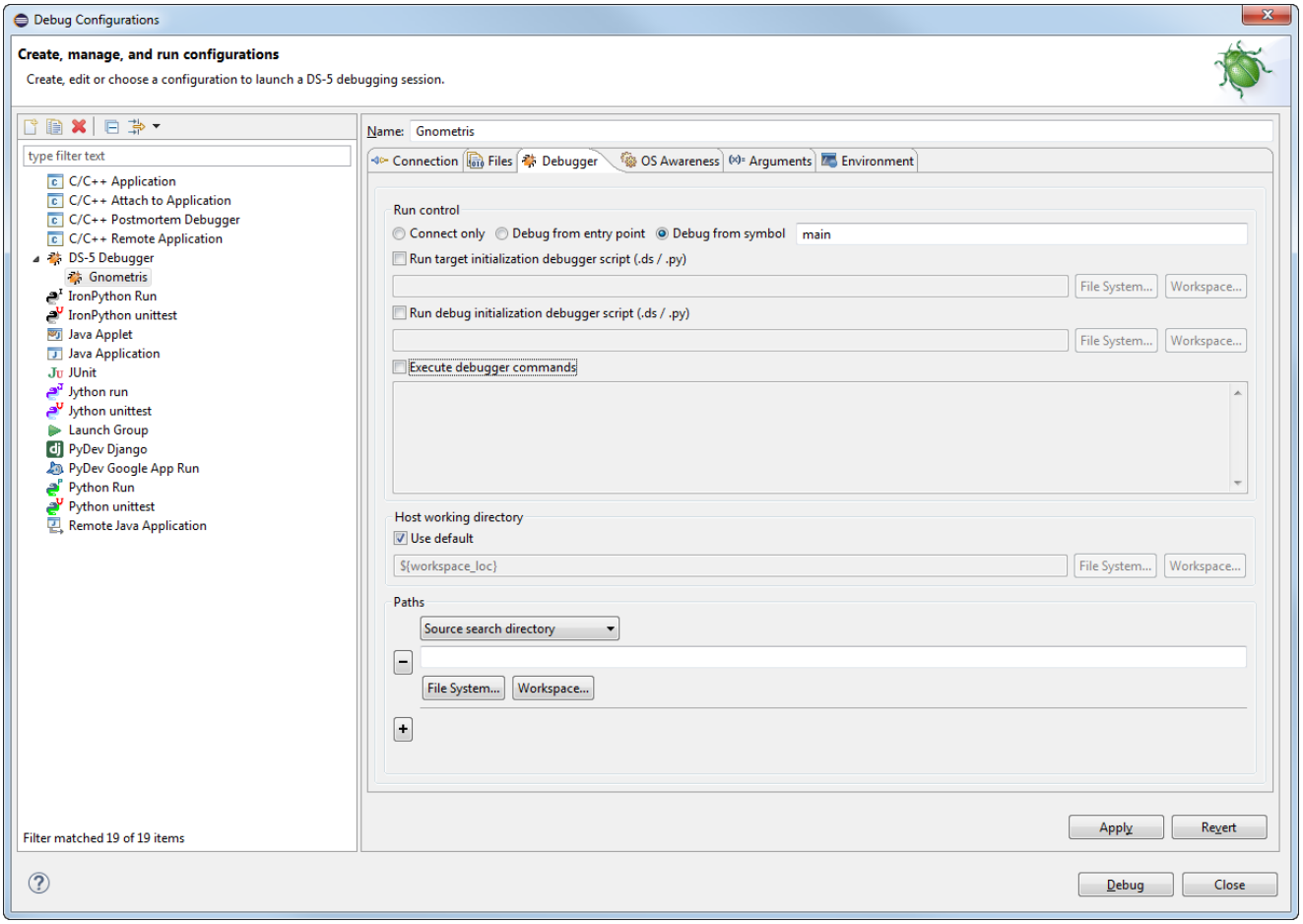



**Figure 4-24 Typical debugger settings for Linux application debug**

8. Click on **Debug** to start the debugger and run to the `main()` function.
9. Debugging requires the DS-5 Debug perspective. If the **Confirm Perspective Switch** dialog box opens, click **Yes** to switch perspective.

## Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.
*4.9 Building the gnometris project from Eclipse* on page 4-64.
*4.10 Building the gnometris project from the command line* on page 4-65.
*4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.
*4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.
*4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.
*4.16 Debugging Gnometris* on page 4-78.

## Related references

*4.2 Examples provided with DS-5* on page 4-46.

## Related information

*Debug Configurations - Connection tab.*
*Debug Configurations - Files tab.*
*Debug Configurations - Debugger tab.*
*Debug Configurations - Environment tab.*

## 4.16 Debugging Gnometris

Debugging the Gnometris application using the example project containing the image binaries and libraries provided with DS-5.

### Procedure

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 Explore**r 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 sourcefile. All the views update as you step through the source code.

**Tip**

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.

### Related tasks

*4.3 Importing the example projects into Eclipse* on page 4-48.
*4.9 Building the gnometris project from Eclipse* on page 4-64.
*4.10 Building the gnometris project from the command line* on page 4-65.
*4.11 Loading the Gnometris application on a Fixed Virtual Platform (FVP)* on page 4-66.
*4.12 Loading the Gnometris application on to an ARM® Linux target* on page 4-67.
*4.13 Configuring an RSE connection to work with an ARM® Linux target* on page 4-68.
*4.15 Connecting to the Gnometris application that is already running on an ARM® Linux target* on page 4-75.

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*C/C++ editor.*
*Debug Control view.*
*Registers view.*

## 4.17 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.

This tutorial uses a simple character device driver `modex.c` which is part of the ARMv7 Linux application examples available in DS-5.

You can use `modex.c` to compile, run, and debug against your target. The `readme.html` in the *`DS-5_install_directory`*`/examples/docs/kernel_module` contains information about customizing this for your target.

**Note**

If you are working with your own module, before you can debug it, you must ensure that you:

- Unpack kernel source code and compile the kernel against exactly the same kernel version as your target.
- Compile the loadable module against exactly the same kernel version as your target.
- Ensure that you compile both images with debug information. The debugger requires run-time information from both images when debugging the module.

### Procedure

1. Create a new **Debug Configuration**.
   a. From the main DS-5 menu, select **Run** > **Debug Configurations**.
   b. In the **Debug Configurations** dialog box, create a **New Launch Configuration** and give it a name. For example, `my_board`.
   c. In the **Connection** tab, select the target and platform and set up your target connection.

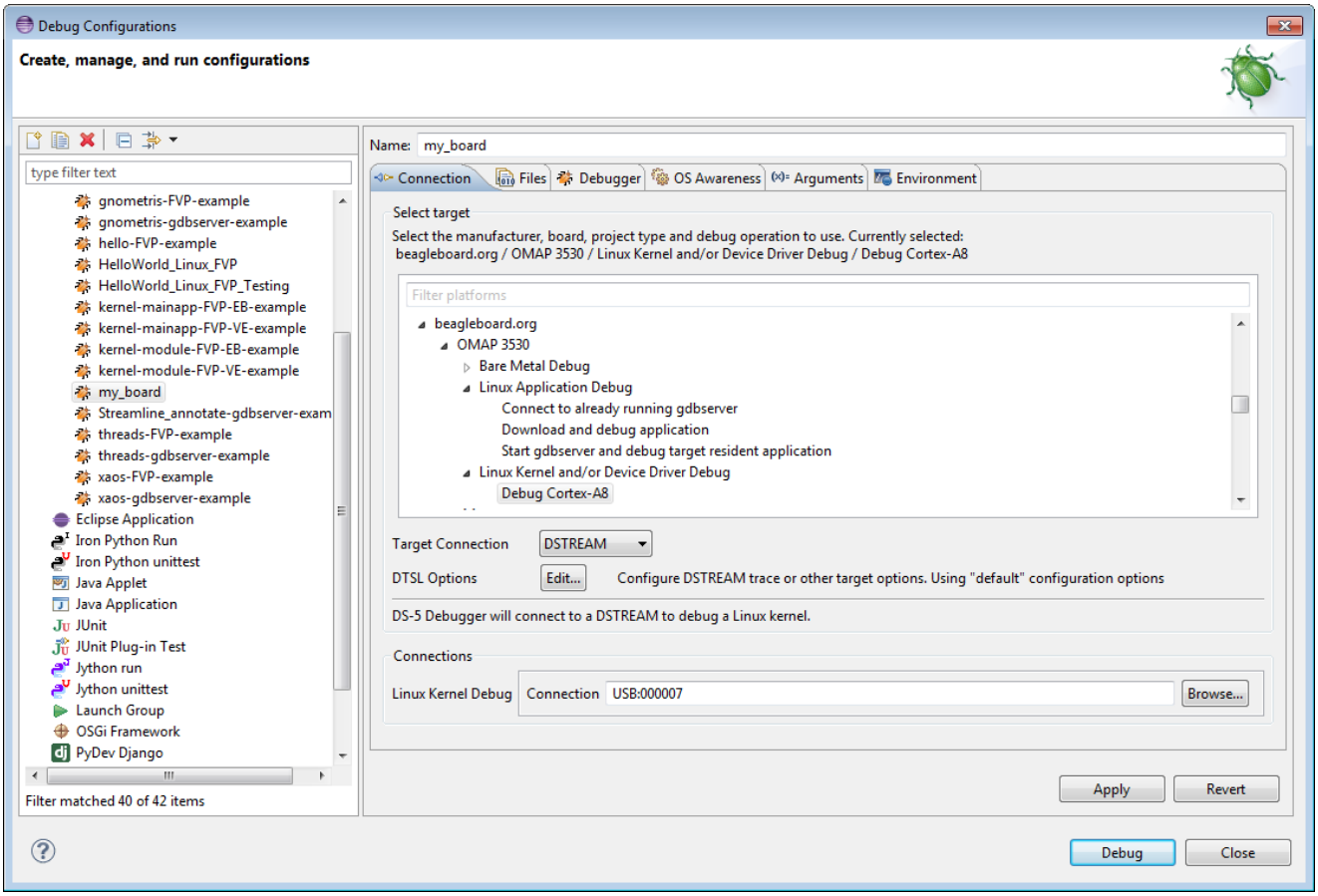



**Figure 4-25 Typical connection settings for a Linux kernel/Device Driver Debug**

d. In the **Files** tab, set up the debugger settings to load debug information for the Linux kernel and the module.

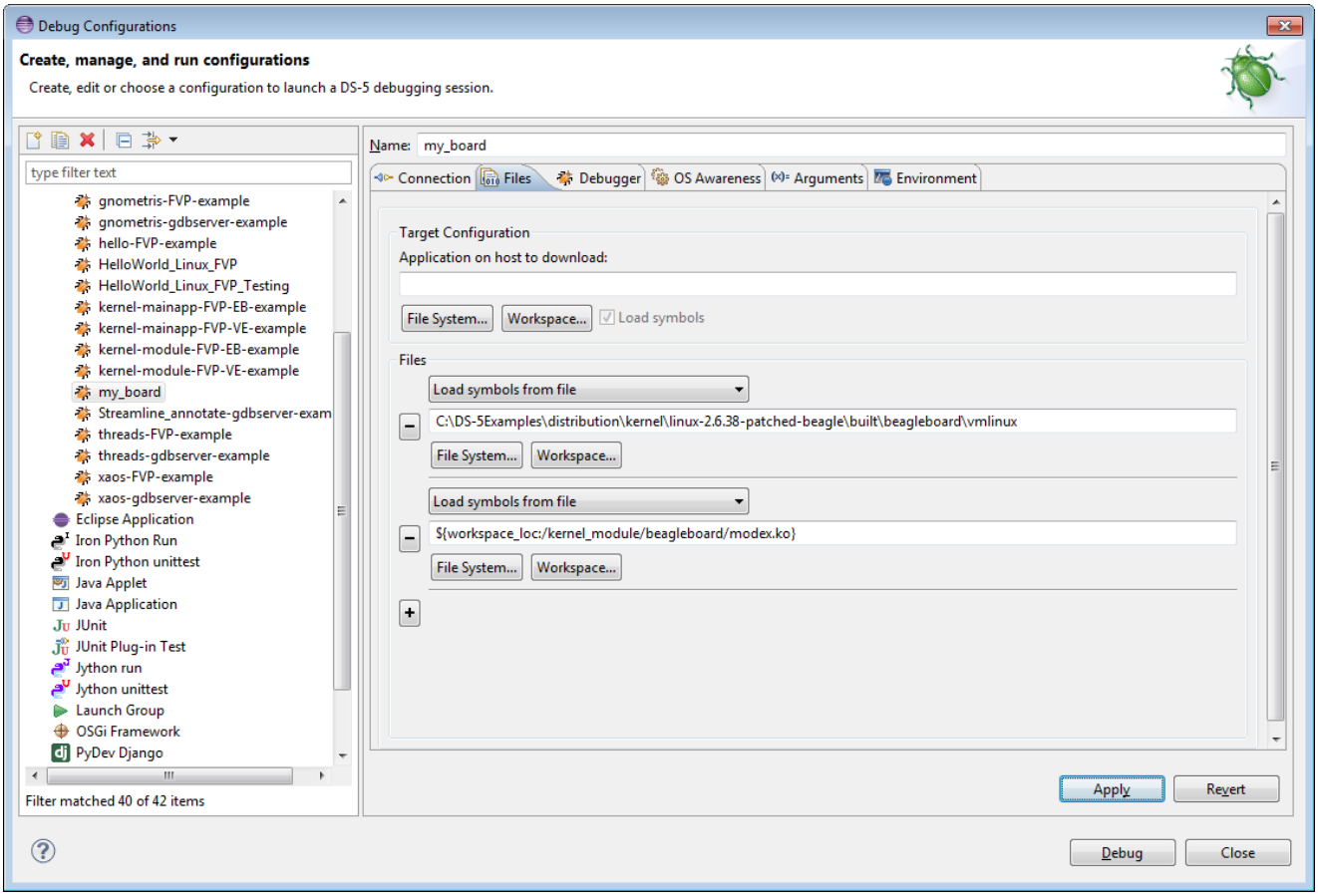



**Figure 4-26 Typical Files settings for a Linux kernel/Device Driver Debug**

e. In the **Debugger** tab, select **Connect only** in the **Run control** panel.
f. Click **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. On the host workstation, navigate to `.../linux_system/kernel_module/stripped/modex.ko` file.
   b. Drag and drop the module to a writeable directory on the target.
4. Using the terminal shell, insert the `modex.ko` kernel module.
   a. Navigate to the location of the kernel 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 using commands as required in the terminal shell. For example: `rmmod modex`
   b. Recompile the module.
   c. Repeat steps *3* to *5* as required.

**Note**

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

Useful terminal shell commands:

`lsmod`
: Displays information about all the loaded modules.

`insmod`
: Inserts a loadable module.

`rmmod`
: Removes a module.

Useful DS-5 Debugger commands:

`info os-modules`
: Displays a list of OS modules loaded after connection.

`info os-log`
: Displays the contents of the OS log buffer.

`info os-version`
: Displays the version of the OS.

`info processes`
: Displays a list of processes showing ID, current state and related stack frame information.

`set os-log-capture`
: Controls the capturing and printing of *Operating System* (OS) logging messages to the console.

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

### Related references

*4.2 Examples provided with DS-5* on page 4-46.

### Related information

*Configuring a connection to a Linux Kernel.*
*Controlling execution.*
*Examining the target.*
*About debugging a Linux kernel.*
*About debugging Linux kernel modules.*
*ARM Linux problems and solutions.*
*Target connection problems and solutions.*
*Operating System (OS) DS-5 debugger commands.*
*Target management terminal for serial and SSH connections.*

## 4.18 Performance analysis of the threads application running on ARM® Linux

ARM Streamline is a graphical performance analysis tool. It captures a wide variety of statistics about code running on the target and uses them to generate analysis reports. You can use these to identify problem areas at system, process, and thread level, in addition to hot spots in the code.

### Prerequisites

This tutorial uses the `threads` example application to show how to use Streamline to capture and analyze profiling data from a Linux target. `threads` is one of the ARMv7 Linux application examples that are provided with DS-5. Before capturing the data, ensure that:

1. You have built the `threads` application.
2. You know the IP address or network name of the target. To find the IP address, you can use the `ifconfig` application in a Linux console. The IP address is denoted by the **inet addr**.
3. The Linux kernel on the target is configured to work with Streamline.
4. The gator daemon, `gatord`, is running on the target. If not, the simplest way to install and run `gatord` on the target is to use the **Setup Target...** button in the **Connection Browser** dialog in the **Streamline Data** view.
5. SSH and `gdbserver` are running on the target.

**Note**

- For more information about building and running the `threads` application on a Linux target see the `readme.html` supplied in the same directory as the source code for the example.
- For more information about how to configure your target for Streamline, see the Streamline User Guide.

### Procedure

1. Launch Eclipse for DS-5 and open the **DS-5 Debug** perspective.
2. In the **Remote Systems** view, define a connection to the target using the **Define a connection to remote system** button.
3. Launch the **Streamline** application.
4. Specify the IP address or network name of the target in the **Address** field. Alternatively, use the **Browse for a target** button, as shown in the following screenshot:

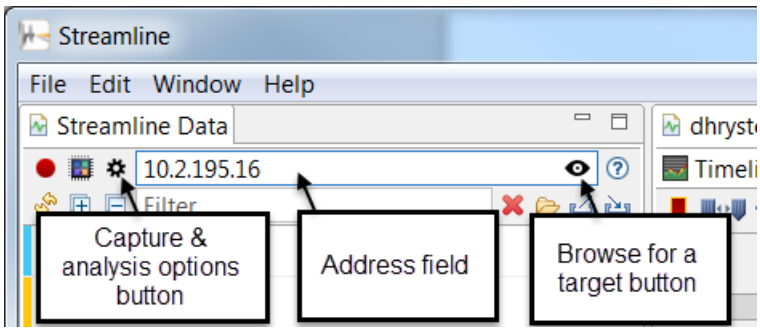



**Figure 4-27 Streamline Data view**

5. Click the **Capture & analysis options** button. In the **Program Images** section, select the `threads` image from the workspace, then select **Save**.
6. In Eclipse for DS-5, select **Run** > **Debug configurations...** then select the `threads-gdbserver` debug configuration. This configuration downloads the application to the target, starts `gdbserver` on the target and starts executing the application, stopping at `main()`.
7. Connect to the target either by clicking on **Debug** in the **Debug Configurations** dialog, or by right clicking on the connection in the **Debug Control** and selecting **Connect to target**.

8. The program stops at `main()`. To start capturing data, switch to the **Streamline** application and click the **Start capture** button . Give the capture file a unique name. The **Live** view opens in **Streamline**, displaying the capture data in real time.
9. In Eclipse for DS-5, press **Continue** to continue executing the code.
10. When the application terminates, stop the capture in Streamline by clicking the **Stop capture and analyze** button .

Streamline automatically analyzes the capture data and produces a report, which it displays in the **Timeline** view, as shown in the following screenshot:

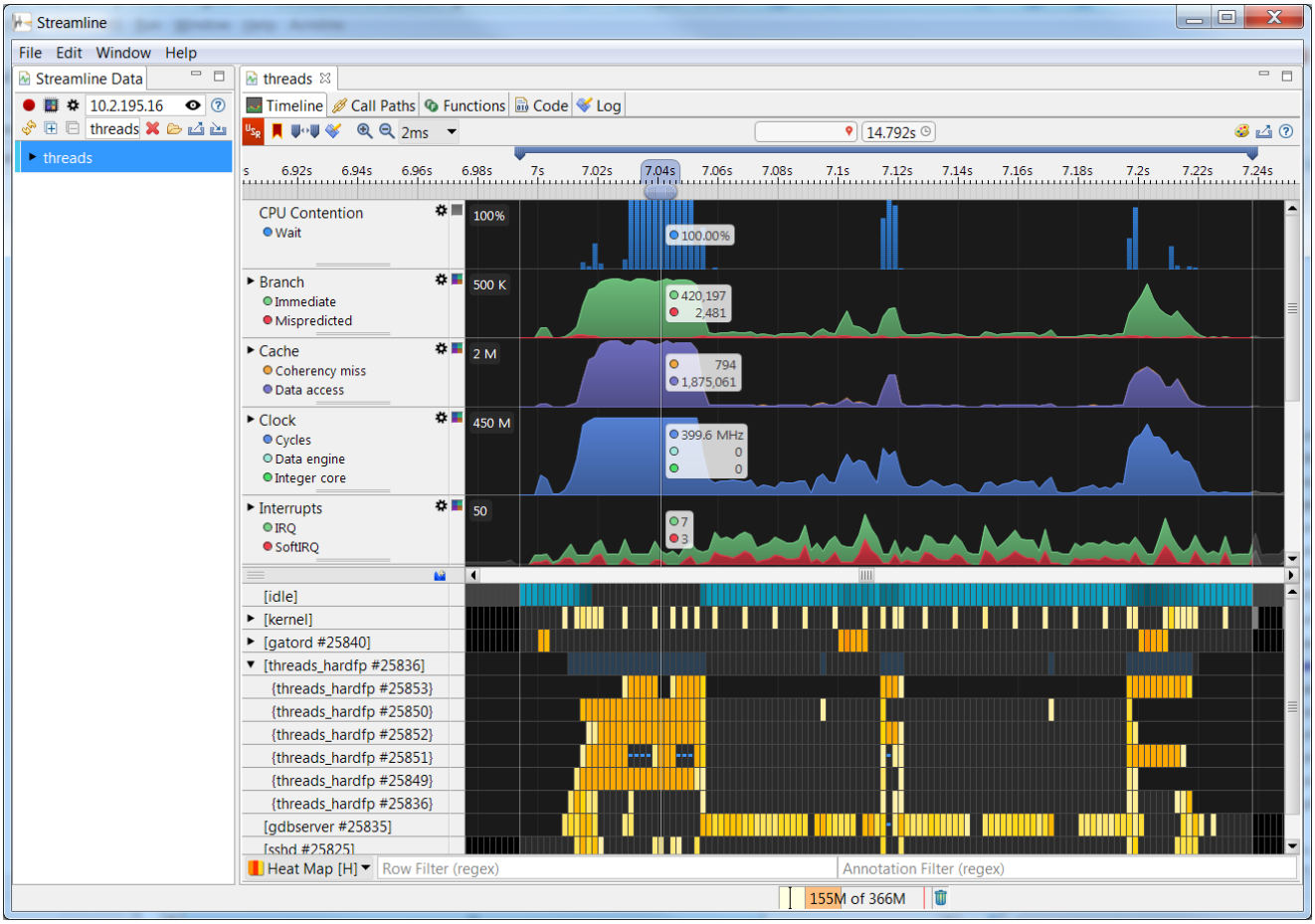



**Figure 4-28 Streamline analysis report for the threads application**

## Related references

*4.2 Examples provided with DS-5* on page 4-46.

## Related information

*Streamline User Guide.*

## 4.19 About registering a new compiler toolchain

You can use a different compiler toolchain other than the one installed with DS-5.

If you want to build projects using a toolchain that is not installed with DS-5, you must first register the toolchain you want to use. You can register toolchains:

- Using the **Preferences** dialog in Eclipse for DS-5.
- Using the `add_toolchain` utility from the DS-5 Command Prompt.

You might want to register a compiler toolchain if:

- You upgrade your version of DS-5 but you want to use an earlier version of the toolchain that was previously installed.
- You install a newer version or older version of the toolchain without re-installing DS-5.

When you register a toolchain, the toolchain is available for new and existing projects in DS-5.

——— **Note** ———

You can only register ARM or GCC toolchains.

———————————

### Related tasks

*4.22 Registering a compiler toolchain from Eclipse* on page 4-91.
*4.20 Registering a compiler toolchain from the DS-5 command prompt* on page 4-86.

## 4.20 Registering a compiler toolchain from the DS-5 command prompt

Use the `add_toolchain` utility from the command prompt to register a new toolchain.

To register a toolchain using the DS-5 command prompt:

### Procedure

1. Enter `add_toolchain` *`path`*, where *`path`* is the directory containing the toolchain binaries. The utility automatically detects the toolchain properties.

   **Note**

   By default, the `add_toolchain` utility is an interactive tool. To use the `add_toochain` utility as a non-interactive tool, add the `--non-interactive` option to the command.

   For example: `add_toolchain C:\ARMCC\5.03\26\ds-win-x86_64-rel\bin --non-interactive`

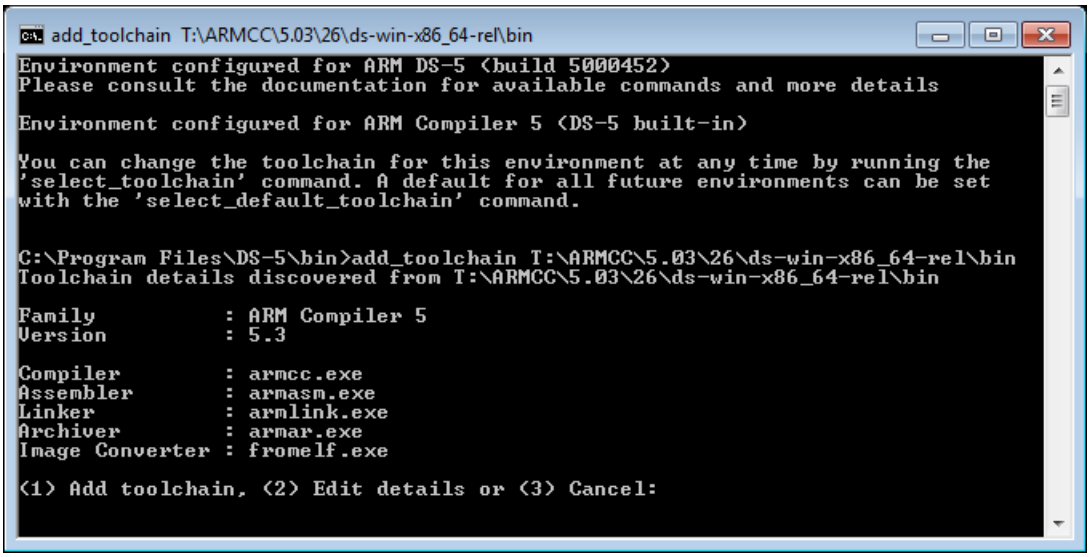



**Figure 4-29 Registering a new toolchain**

2. The utility prompts whether you want to register the toolchain with the details it has detected. If you want to change the details, the utility prompts for the details of the toolchain.

   **Note**

   - The toolchain type must be one of ARM Compiler 4, ARM Compiler 5, ARM Compiler 6, or GCC.
   - The toolchain target only applies to GCC toolchains. It indicates what target platform the GCC toolchain builds for. For example, if your compiler toolchain binary is named `arm-linux-gnueabihf-gcc`, then the target name is the prefix `arm-linux-gnueabihf`. The target field allows DS-5 to distinguish different toolchains that otherwise have the same version.

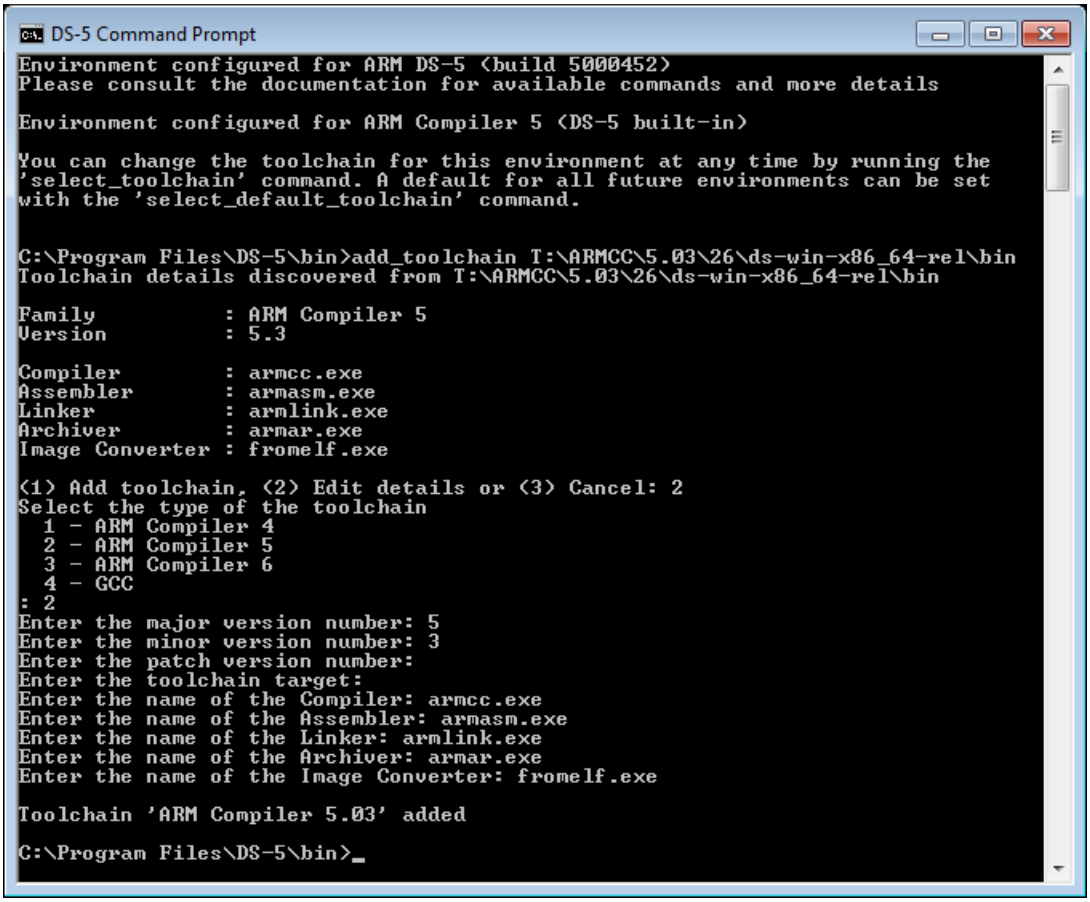



**Figure 4-30 Registering a new toolchain**

**Note**

You must manually enter the toolchain properties if:

- The toolchain properties were not autodetected.
- The type, major version, and minor version of the new toolchain are identical to a toolchain that DS-5 already knows about.

3. After you register a new toolchain, you must restart DS-5 before you can use the toolchain in the DS-5 environment.
4. When you create a new project, DS-5 shows the new toolchain in the available list of toolchains. In this example, `ARMCCv5.01` is the newly registered toolchain.

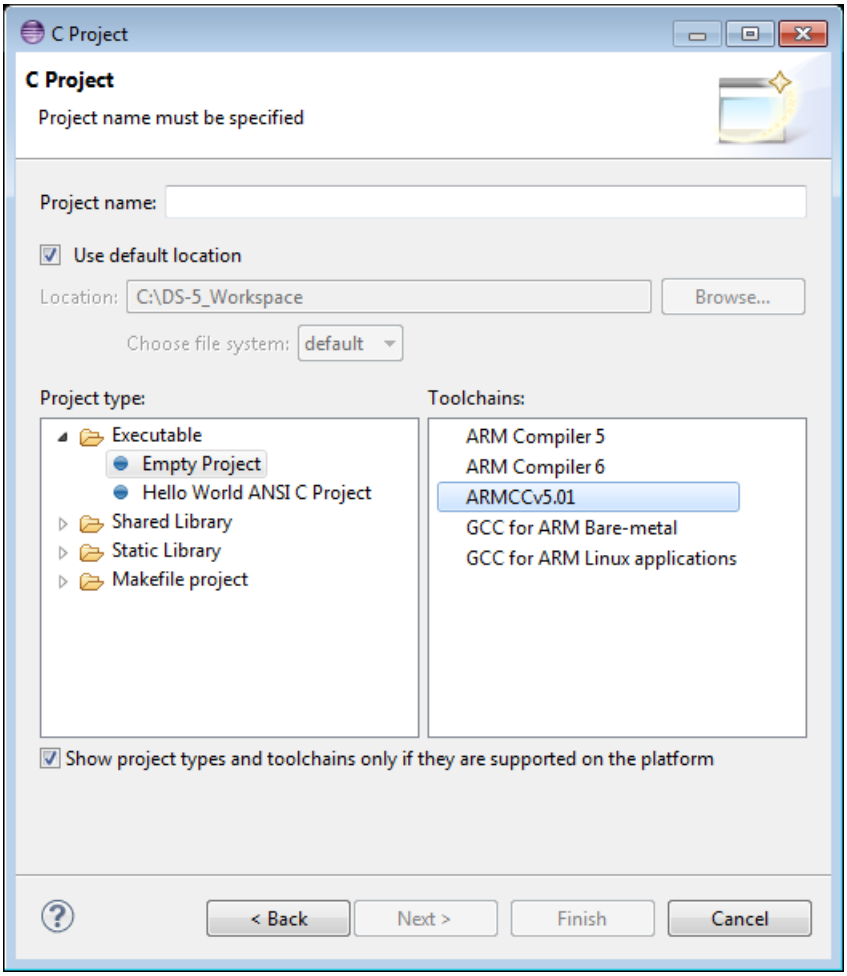



**Figure 4-31 Using a new toolchain for a new project**

For an existing project, if you want to change the toolchain to the newly registered toolchain, use the **Tool Chain Editor** dialog.

- Right-click the project and select **Properties** to show the **Properties** dialog.
- Select **C/C++ Build** > **Tool Chain Editor**

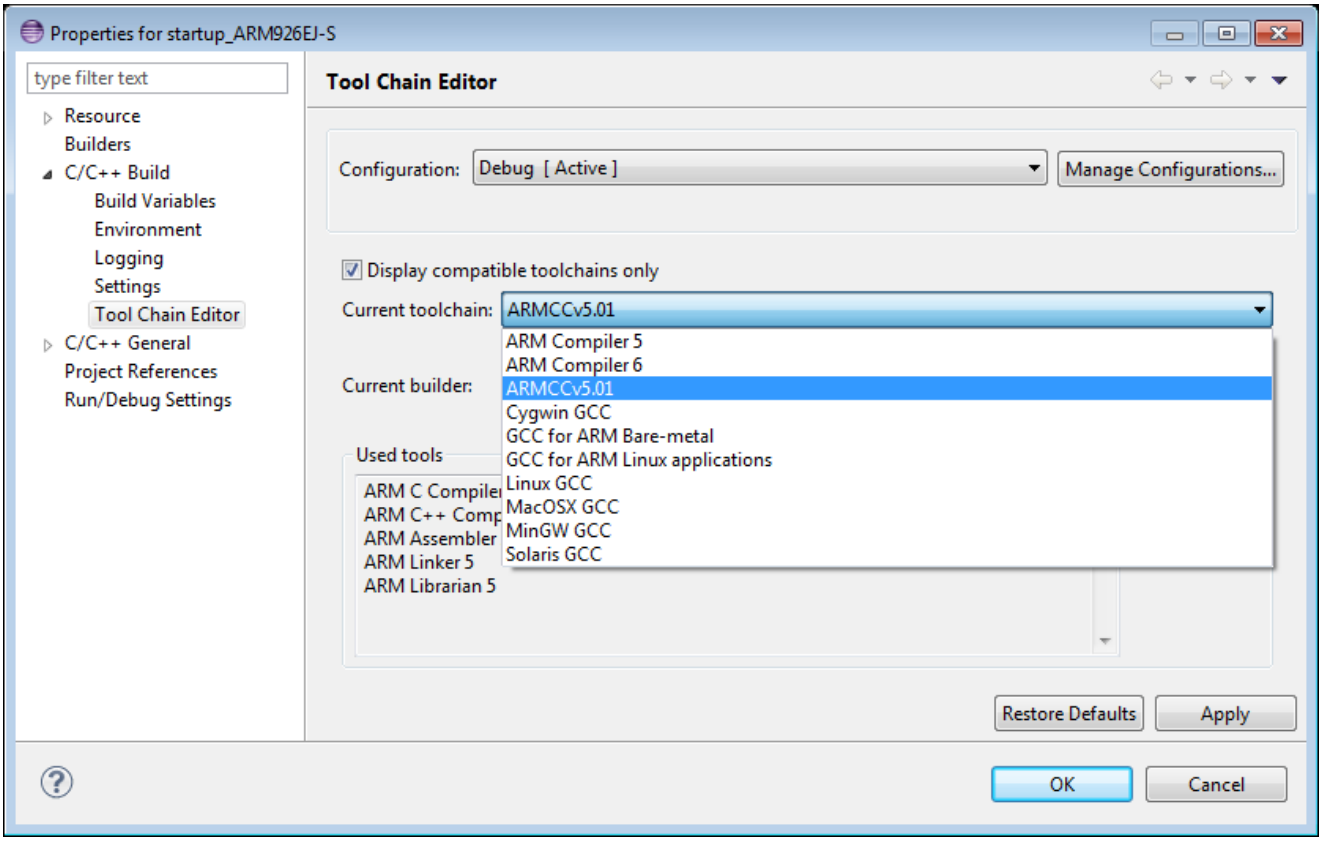



**Figure 4-32 Changing the toolchain for a project**

## 4.21 Configuring a compiler toolchain for the DS-5 command prompt

When you want to compile or build from the DS-5 command prompt, you can select the compiler toolchain you want to use. You can set this as the default toolchain so that you do not need to select a toolchain every time you start the DS-5 command prompt.

**Note**

By default, the DS-5 command prompt is not configured with a compiler toolchain.

On Linux, run `suite_exec` with the `--toolchain` *`name`* option to configure a compiler toolchain for the DS-5 environment. Run `suite_exec` with no arguments for the list of available toolchains.

For example, to use the ARM Compiler 5 toolchain included in DS-5, run:

```
DS-5_install_directory/bin/suite_exec --toolchain "ARM Compiler 5 (DS-5 built-in)"
bash --norc
```

You can also run the command `set_default_toolchain` and follow the instructions to select the default compiler for the command prompt.

On Windows, to set a default compiler toolchain for the DS-5 command prompt, use the `select_default_toolchain` command.

The following procedure describes the steps for using `select_default_toolchain` on Windows.

### Procedure

1. Open the DS-5 command prompt, by selecting **Start** > **All Programs** > **DS-5 Command Prompt**.
2. Enter `select_default_toolchain` on the DS-5 command prompt. This lists the available compiler toolchains.

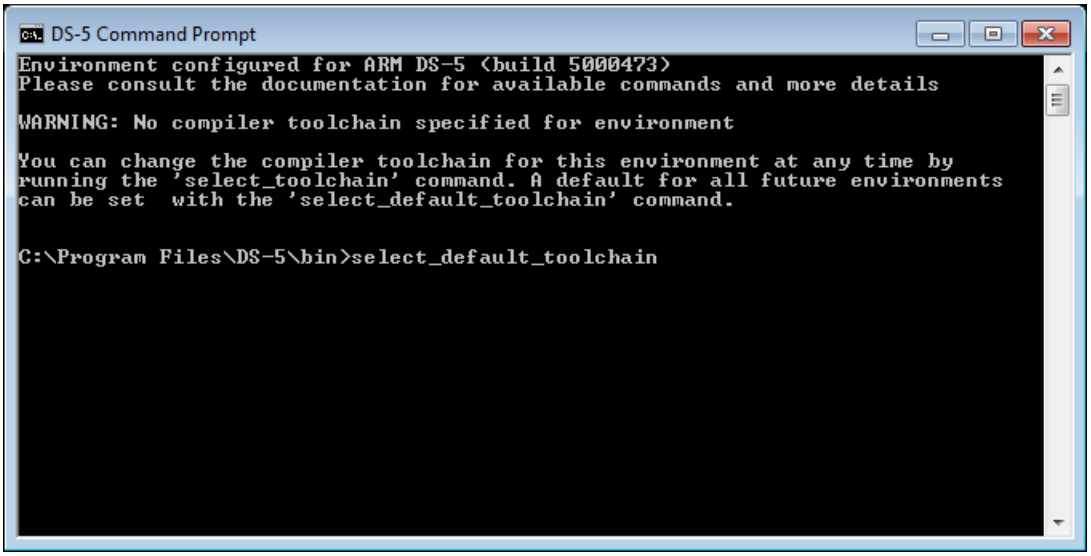



**Figure 4-33 Configuring a default toolchain**

3. Select your default compiler toolchain. For example, enter `1` to select the ARM Compiler 5 that is built-in with DS-5. This configures the DS-5 command prompt for the selected toolchain. When you open a new DS-5 command prompt, the environment is still configured for your selected toolchain.

   **Note**

   To configure a compiler toolchain for the current DS-5 command prompt, without changing the default toolchain, use the `select_toolchain` command.

## 4.22 Registering a compiler toolchain from Eclipse

You can register compiler toolchains using the **Preferences** dialog in Eclipse for DS-5.

### Procedure

1. To view the compiler toolchains that DS-5 currently knows about, select **Windows** > **Preferences**. And then select **DS-5** > **Toolchains**.

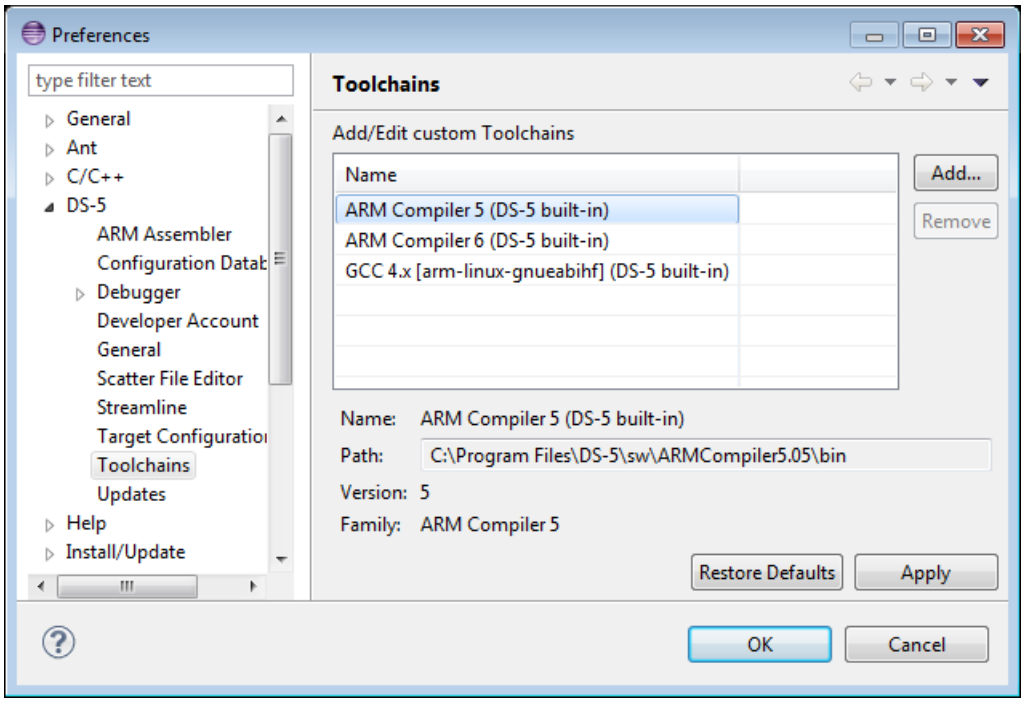



**Figure 4-34 Toolchains Preferences dialog**

2. To add a toolchain, select **Add**. This displays the **Add a new Toolchain** dialog.
3. Enter the path to the toolchain binaries that you want to use. Then click **Next** to autodetect the toolchain properties.
4. When the toolchain properties have been autodetected, you can select **Finish** to register the toolchain. Alternatively, select **Next** to manually enter or change the toolchain properties, and then select **Finish**.

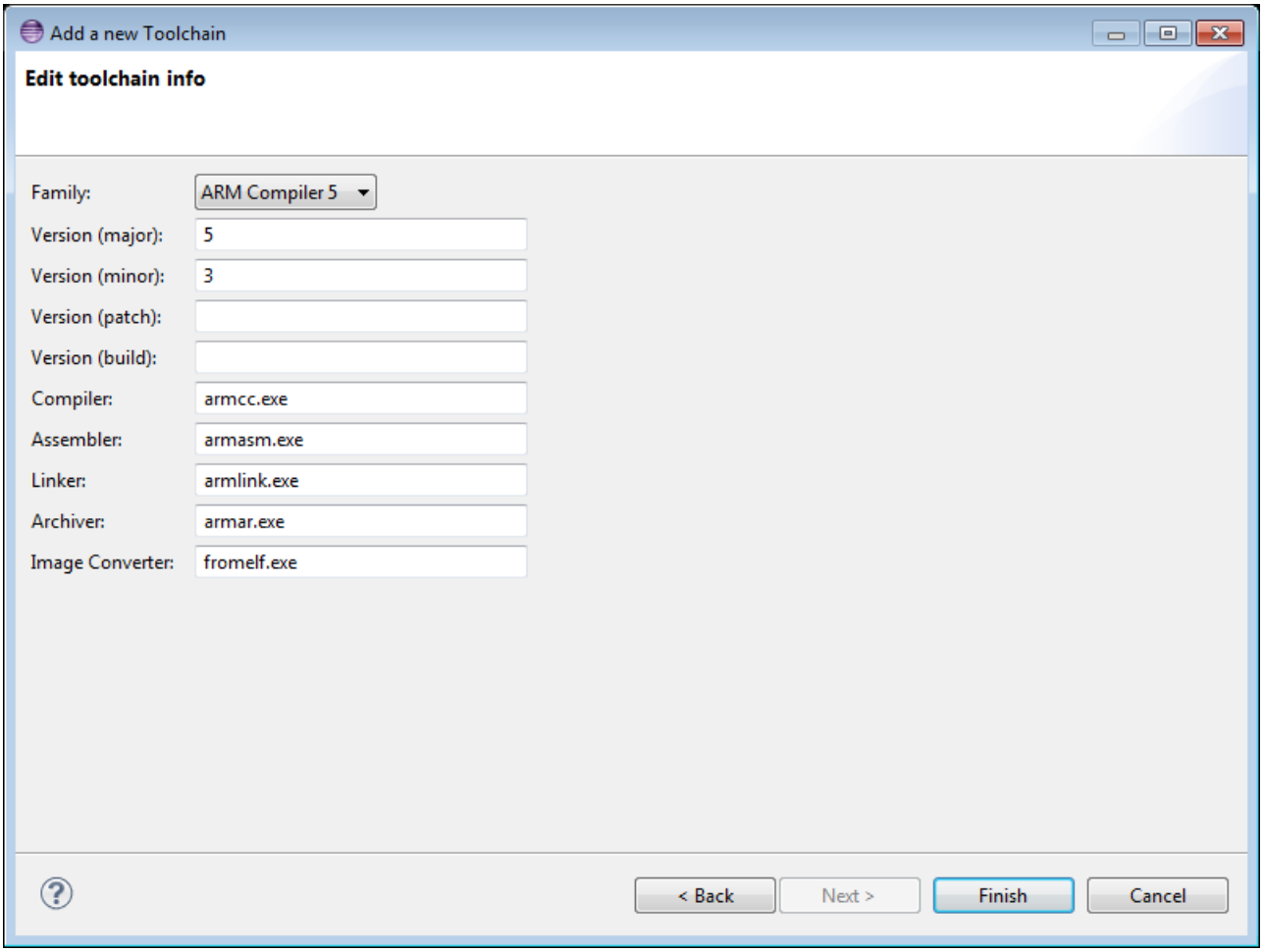



**Figure 4-35 Properties for the new toolchain**

**Note**

You must manually enter the toolchain properties if:

- The toolchain properties were not autodetected.
- The family, major version, and minor version of the new toolchain are identical to a toolchain that DS-5 already knows about.

5. Select **Apply** from the **Toolchains** preferences dialog. The new toolchain has now been registered into DS-5. You must restart DS-5 before you can use the new toolchain in the DS-5 environment.

### Related tasks

*4.20 Registering a compiler toolchain from the DS-5 command prompt* on page 4-86.

# X-ON Electronics

Largest Supplier of Electrical and Electronic Components

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

Other Similar products are found below :

ERD-S1TJ8R2V DP3-22 ECE-A1HKAR47 LC-R063R4P AH64-05846A ELL-ATV100M ERA-14EB121U ECOS1JA122BA ECW-U1C184JB9 HC2-H-AC48V-F ERA-S15J471V HC2-HP-AC115V-F ECJ-2FF1A475Z ECOS2GP271EA EYG-A091210P EEV-HB1HR22R HC4-H-DC12V ELC-12D471E EVM-3RSX50B13 EEF-SD0E221R EVM-1USX30B12 EEF-UE0E471LR EEF-CD0K8R2R EEF-UE0E471R HHR-80AAAB3B ELC-10D330E ERA-V15J101V HC2-SF-K EVQ-PSC02K EEV-TG2A220P 036506R ERD-S1TJ165V LC-P127R2P ECE-V0JA220NR 2SB15990QL RP-SMLE16DA1 ECOS2GP121CX EVM-3VSX50B52 RP-SDME04DA1 ELC-09D4R7F ELJRF22NJFB ELJFCR82KF EEV-HA2A3R3P EVM-F6SA00B55 ESE-15700 EEC-S5R5H105N EEV-TG1J330P AXE260124A EEV-TG2A100P ECJ-1VF1E683Z