Model Debugger for Fast Models
User Guide

Copyright © 2007-2009 ARM Limited. All rights reserved.

Release Information

<table>
<thead>
<tr>
<th>Description</th>
<th>Issue</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 2007</td>
<td>B</td>
<td>Confidential</td>
<td>Updated for System Generator 3.1.</td>
</tr>
<tr>
<td>February 2008</td>
<td>C</td>
<td>Confidential</td>
<td>Updated for System Generator 3.2. Supports RHEL 5. New shortcut keys</td>
</tr>
<tr>
<td>March 2008</td>
<td>D</td>
<td>Confidential</td>
<td>Updated for SoC Designer 7.1.</td>
</tr>
<tr>
<td>June 2008</td>
<td>E</td>
<td>Confidential</td>
<td>Updated for Model Debugger 4.0 as provided with System Generator 4.0</td>
</tr>
<tr>
<td>August 2008</td>
<td>F</td>
<td>Confidential</td>
<td>Update for System Generator 4.0 SP1.</td>
</tr>
<tr>
<td>March 2009</td>
<td>H</td>
<td>Non-Confidential</td>
<td>Update for Model Debugger for Fast Models 4.2.</td>
</tr>
<tr>
<td>April 2009</td>
<td>I</td>
<td>Non-Confidential</td>
<td>Update for Model Debugger for Fast Models 5.0.</td>
</tr>
</tbody>
</table>

Proprietary Notice

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

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

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

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

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

Copyright © 2007-2009 ARM Limited. All rights reserved.

Non-Confidential, Unrestricted Access
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

Model Debugger for Fast Models User Guide

## Preface

About this book .......................................................................................................................... xii  
Feedback ..................................................................................................................................... xiv

## Chapter 1  Introduction

1.1 Overview .......................................................................................................................... 1-2

## Chapter 2  Installation and Configuration

2.1 Linux installation procedure ............................................................................................... 2-2  
2.2 Windows installation procedure ....................................................................................... 2-4

## Chapter 3  Using Model Debugger

3.1 Launching Model Debugger .............................................................................................. 3-2  
3.2 Model Debugger application windows ............................................................................... 3-11  
3.3 Debug views for source code and disassembly ................................................................. 3-27  
3.4 Debug views for registers and memory ............................................................................ 3-37  
3.5 Debug views for pipelines ............................................................................................... 3-44  
3.6 Watch window and Expression Evaluator ....................................................................... 3-50  
3.7 Setting breakpoints in the debug views ............................................................................ 3-54  
3.8 Model Debugger sessions ............................................................................................... 3-59  
3.9 Preferences dialog ........................................................................................................... 3-60
Appendix A  Shortcuts
A.1  Keyboard shortcuts ..................................................................................... A-2
# List of Tables

Model Debugger for Fast Models User Guide

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Environment variables</td>
<td>2-3</td>
</tr>
<tr>
<td>3-1</td>
<td>Command line options</td>
<td>3-2</td>
</tr>
<tr>
<td>3-2</td>
<td>Operator precedence</td>
<td>3-52</td>
</tr>
<tr>
<td>A-1</td>
<td>Keyboard shortcuts</td>
<td>A-2</td>
</tr>
</tbody>
</table>
List of Figures

Model Debugger for Fast Models User Guide

Figure 1-1 Model Debugger main window with debug windows ................................................. 1-3
Figure 3-1 Launching Model Debugger using the System Canvas toolbar ................................ 3-6
Figure 3-2 System Canvas Configure Model Parameters dialog ............................................... 3-6
Figure 3-3 System Canvas Configure Model Parameters dialog, list view .............................. 3-7
Figure 3-4 Select Targets dialog ............................................................................................... 3-8
Figure 3-5 Load Application dialog .......................................................................................... 3-9
Figure 3-6 Connect remote dialog ............................................................................................. 3-10
Figure 3-7 Select Target dialog ................................................................................................ 3-10
Figure 3-8 Default layout for Model Debugger ......................................................................... 3-12
Figure 3-9 Main toolbar .......................................................................................................... 3-12
Figure 3-10 Drag-and-drop of debug views ................................................................................ 3-21
Figure 3-11 Duplicating a register view ..................................................................................... 3-22
Figure 3-12 Layout Control window ......................................................................................... 3-23
Figure 3-13 Layout Control context menu ................................................................................ 3-23
Figure 3-14 Load Layout dialog ............................................................................................... 3-24
Figure 3-15 Icons for selecting a new debug view .................................................................... 3-25
Figure 3-16 Closing windows or individual debug views ......................................................... 3-25
Figure 3-17 Arrow button for scrolling code .......................................................................... 3-27
Figure 3-18 Source view ........................................................................................................ 3-28
Figure 3-19 Debug Source Files dialog .................................................................................... 3-29
Figure 3-20 Find Source File dialog ....................................................................................... 3-30
Figure 3-21 Source File Properties dialog ............................................................................... 3-30
Figure 3-22 Source Path Replacement dialog ........................................................................ 3-31
List of Figures

Figure 3-23 Find dialog ......................................................................................................... 3-33
Figure 3-24 Disassembly view ............................................................................................. 3-33
Figure 3-25 Matching source and disassembly ................................................................. 3-34
Figure 3-26 Call Stack view ............................................................................................... 3-35
Figure 3-27 Select register group ....................................................................................... 3-37
Figure 3-28 Register view showing current and previous contents ..................................... 3-37
Figure 3-29 Register view contents at cursor ....................................................................... 3-38
Figure 3-30 Memory view ................................................................................................. 3-39
Figure 3-31 Local Variable view ......................................................................................... 3-42
Figure 3-32 Global Variable view ....................................................................................... 3-42
Figure 3-33 Pipeline Overview window ............................................................................... 3-44
Figure 3-34 Pipeline Table window .................................................................................... 3-45
Figure 3-35 Pipeline Table icons ........................................................................................ 3-45
Figure 3-36 Pipeline Table context menu ............................................................................ 3-46
Figure 3-37 Pipeline Stage Properties dialog ..................................................................... 3-47
Figure 3-38 Pipeline view context menu ............................................................................. 3-48
Figure 3-39 Submenu for display format ............................................................................ 3-49
Figure 3-40 Watch window .................................................................................................. 3-50
Figure 3-41 Source view breakpoint .................................................................................... 3-54
Figure 3-42 Disassembly view breakpoint ......................................................................... 3-54
Figure 3-43 Register view breakpoint ............................................................................... 3-55
Figure 3-44 Pipeline table breakpoint ................................................................................ 3-56
Figure 3-45 Breakpoint Manager dialog ............................................................................ 3-57
Figure 3-46 Breakpoint Properties dialog ......................................................................... 3-58
Figure 3-47 Preferences dialog .......................................................................................... 3-60
Preface

This preface introduces the Model Debugger for Fast Models User Guide. It contains the following sections:

- *About this book* on page xii
- *Feedback* on page xiv.
About this book

This book describes the operation of the Model Debugger for Fast Models.

Intended audience

This book has been written for experienced hardware and software developers to aid the development of processor architectures using Fast Model Tools design tools such as System Generator for Fast Models.

Model Debugger connects to any Cycle Accurate Debug Interface (CADI) compliant model that is either:

• running standalone
• embedded into a multi-core system simulation in, for example, System Canvas.

Organization

This book is organized into the following chapters:

Chapter 1 Introduction
Read this chapter for an introduction to Model Debugger.

Chapter 2 Installation and Configuration
Read this chapter for details on how to install and configure Model Debugger. If you have already installed System Generator, Model Debugger is already installed for you so you can safely ignore this chapter.

Chapter 3 Using Model Debugger
Read this chapter to learn how to use Model Debugger.

Appendix A Shortcuts
Refer to this appendix for a list of keyboard shortcuts available in Model Debugger.

Typographical conventions

The typographical conventions are:

italic Highlights important notes, introduces special terminology, denotes internal 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.

**monospace** 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 >** Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example:

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

**Further reading**

This section lists related publications by ARM.

See http://infocenter.arm.com for access to ARM documentation.

**ARM publications**

This book contains information that is specific to this product. See the following documents for other relevant information:

- *Cycle Accurate Debug Interface Developer Guide* (ARM DUI 0444)
- *ARM Architecture Reference Manual* (ARM DDI 0100)
Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

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

• the product name
• a concise explanation.

Feedback on this book

If you have any comments on this book, send an e-mail to errata@arm.com. Give:

• the title
• the number
• the relevant page number(s) to which your comments apply
• a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.
Chapter 1

Introduction

This chapter describes the main features of Model Debugger. It contains the following section:

• Overview on page 1-2.
1.1 Overview

Model Debugger for Fast Models is a fully retargetable debugger for scalable multicore software development. It is specifically designed to address the requirements of SoC software developers. Model Debugger has an easy to use GUI front end and supports:

- source-level debugging
- complex breakpoints
- advanced symbolic register display
- customized window layout.

Figure 1-1 on page 1-3 shows an example of a Model Debugger session.

Model Debugger can connect to any Cycle Accurate Debug Interface (CADI) compliant models. These models can be either:

- embedded into a multi-core system simulation such as in System Canvas
- run standalone.

For more information on the CADI interface, see the ARM Cycle Accurate Debug Interface Developer’s Guide.

Full multi-core debugging support is included and multiple instances of Model Debugger stay fully synchronized while debugging different cores running within a single system.
1.1.1 Key features

The key features of Model Debugger are:

- full simulation control on C-statement, instruction and cycle levels
- C-source level display with syntax highlighting
- integrated variable browser
- low-level disassembly display
- call stack and backtrace
- complex register display with unlimited register groups and compound registers
- memory windows with support for multiple memory spaces and bit widths
- breakpoints on register and memory locations with complex conditions
- save and restore support for the model state
- step-back support
- hardware multi-threading support with context awareness
- full cycle awareness including detailed pipeline display
- advanced search capabilities
• intuitive GUI with fully customizable window layout
• project management to store debugging sessions including window layout, open files and breakpoints.

1.1.2 Retargetable debugger

Model Debugger supports completely retargetable debugging of any target that supports the CADI debug interface. All target-related information, such as the disassembly and resources like register files and flags, is contained in the target model library. Model Debugger uses the CADI debug interface to communicate with the target, determine the target state, and control execution. See the ARM Cycle Accurate Debug Interface Developer’s Guide.

Model Debugger can attach to and debug target components that are part of Fast Model systems. It can also be used to debug any stand-alone target model library that has a CADI interface.

1.1.3 Multi-core debugging

Model Debugger provides specific support for multi-core debugging and can be attached to an arbitrary number of processor targets in a multi-core system.

If launched from System Canvas and attached to a core model, Model Debugger automatically loads the debug information for the respective target processor and colors all views.

Model Debugger can save the appearance for each target based on project files. Information that can be saved and restored includes:
• debugger geometry
• complete layout and geometry of all views
• breakpoints.

For a detailed description of Model Debugger project files, see Model Debugger sessions on page 3-59.
Chapter 2
Installation and Configuration

This chapter describes how to install and configure Model Debugger. It contains the following sections:
• Linux installation procedure on page 2-2
• Windows installation procedure on page 2-4.

——— Note ————
The installation and configuration steps described in this chapter are only required if you are installing a standalone version of Model Debugger.

Model Debugger is automatically installed when you install Fast Models so you can ignore this chapter if you use the Fast Model tools such as System Canvas for Fast Models.

——— ————
2.1 Linux installation procedure

This section describes the procedure for installing Model Debugger on Linux.

2.1.1 Software requirements

Model Debugger requires the following software components to be installed in order to run correctly:

**Host system**  Red Hat Enterprise Linux version 4 or 5.
Adobe Acrobat reader.

**License management utilities**

FLEXlm version 9.2 or higher.

The FLEXlm (or FLEXnet, for version 10.8) license management utilities are provided separately on the web site from which you downloaded the Model Debugger tools. Use the highest version of the license management utilities provided with any ARM® tools you are using. If you do not have a copy of these utilities, contact ARM License Support, license.support@arm.com.

2.1.2 Installation

Model Debugger is installed by unpacking the archive and running the setup program as shown:

```
 gunzip ModelDebugger_version.tgz
tar -xvf ModelDebugger_version.tar
./setup.bin
```

In the sequence of commands, *version* is the version of Model Debugger you are installing.

The installer prompts you for the target installation directory and creates the following subdirectories:

- **bin**  executables
- **doc**  documentation
- **etc**  Model Debugger setup scripts
- **lib**  libraries and tool-specific files.
2.1.3 Configuring your environment

Model Debugger provides setup scripts in the `etc` directory. The appropriate setup script must be executed to configure your environment for Model Debugger:

- For Bourne and related shells use:
  
  ```bash
  . $installation_directory/etc/setup.sh
  ```

- For C and related shells use:
  
  ```bash
  source $installation_directory/etc/setup.csh
  ```

You might find it more convenient to add a reference to the Model Debugger configuration script to your usual startup script.

The setup script sets the environment variables listed in Table 2-1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PATH</td>
<td>The <code>PATH</code> environment variable is updated to include <code>installation_directory/bin</code>.</td>
</tr>
<tr>
<td>$MAXVIEW_HOME</td>
<td>Set to the Model Debugger installation directory. Model Debugger was previously named MaxView.</td>
</tr>
<tr>
<td>$ARMLMD_LICENSE_FILE</td>
<td>Set to the location of the license file or license server (using <code>port@host</code> syntax) for Model Debugger. See <em>ARM FLEXnet License Management Guide</em>. If necessary you can change this environment variable after installation by editing the <code>setup.[c]sh</code> script.</td>
</tr>
</tbody>
</table>
2.2 Windows installation procedure

This section describes the procedure for installing Model Debugger on Windows.

2.2.1 Software requirements

Model Debugger requires the following software components to be installed in order to run correctly:

**Host system**  Windows XP or Vista.
Adobe Acrobat reader.

**License management utilities**

FLEX\lm version 9.2 or higher.

The FLEX\lm (or FLEX\net, for version 10.8) license management utilities are provided separately on the web site from which you downloaded the Model Debugger tools. Use the highest version of the license management utilities provided with any ARM tools you are using. If you do not have a copy of these utilities, contact ARM License Support, license.support@arm.com.

2.2.2 Installation

To install Model Debugger:

1. Open the distribution archive ModelDebugger\_version.zip and extract the complete contents into a temporary directory. \textit{version} is the version of Model Debugger you are installing.

2. Run the Setup.exe program in the temporary directory to start the installer.

3. When prompted by the installer, specify the target directory for the installation or accept the default directory.

4. The installer creates the following subdirectories in the specified installation directory:
   - \texttt{bin} executables
   - \texttt{doc} documentation.

   **Note**

   The installer configures the environment variables for the user who installed the tools. If it is necessary that other users on the same computer can use Model Debugger, you must either copy the value of the %MAXVIEW\_HOME% environment variable to their system or create a new value for the environment variable.
variable to the System variables, or get the other users to define the environment variable themselves in Control Panel. You must have administrator privileges to perform either of these operations.
Chapter 3
Using Model Debugger

This chapter describes how to use Model Debugger. It contains the following sections:

• Launching Model Debugger on page 3-2
• Model Debugger application windows on page 3-11
• Debug views for source code and disassembly on page 3-27
• Debug views for registers and memory on page 3-37
• Debug views for pipelines on page 3-44
• Watch window and Expression Evaluator on page 3-50
• Setting breakpoints in the debug views on page 3-54
• Model Debugger sessions on page 3-59
• Preferences dialog on page 3-60.

——— Note ————
There is typically no difference, other than the appearance of windows and icons, between the Linux and Windows version of Model Debugger. Unless otherwise indicated, information in this section applies to both versions of Model Debugger.
3.1 Launching Model Debugger

Model Debugger can be launched from the command line or System Canvas. You can also start Model Debugger independently of these GUI tools if required.

Use the Preferences dialog to modify the preferences of Model Debugger after it has started. See Preferences dialog on page 3-60.

3.1.1 Using the command line

To run Model Debugger from the command line, type `modeldebugger`.

Table 3-1 lists the arguments and options that can be passed on the command line.

<table>
<thead>
<tr>
<th>Short</th>
<th>Long option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--nogui</td>
<td>--nogui</td>
<td>Run the simulation without displaying the GUI. See Running Model Debugger without a GUI on page 3-5. Use the <code>--script</code> option to load and run a script on startup.</td>
</tr>
<tr>
<td></td>
<td>--cyclelimit cycles</td>
<td>Set a limit on the number of system cycles for a simulation in non-GUI mode. See Running Model Debugger without a GUI on page 3-5. The cyclelimit option is only enabled if the <code>--nogui</code> option is used.</td>
</tr>
<tr>
<td></td>
<td>--timelimit time</td>
<td>Set a time limit for a simulation in non-GUI mode. See Running Model Debugger without a GUI on page 3-5. The timelimit option is only enabled if the <code>--nogui</code> option is used.</td>
</tr>
<tr>
<td>-a</td>
<td>--application filename</td>
<td>Load the application file <code>filename</code>. To target cores in multicore systems, prefix the name with the path to the instance. For example, <code>foo.bar.core=dhrystone.axf</code>. See also String syntax on page 3-4.</td>
</tr>
<tr>
<td>-C</td>
<td>--parameter parameter</td>
<td>Set a single parameter of the model. Parameters are specified as a path naming the instance and the parameter name using dot separators. For example, <code>foo.bar.inst.parameter=1000</code>. If it is necessary to set multiple parameters at once, use the <code>--config-file</code> option instead.</td>
</tr>
<tr>
<td>-c</td>
<td>--connect simulation_id</td>
<td>Connect to a remote CADI simulation. The simulation to connect to is specified by the <code>simulation_id</code>. Use the <code>--list-connections</code> option to display the list of available connections.</td>
</tr>
<tr>
<td>-E</td>
<td>--enable-verbose msgClass</td>
<td>Use verbose messages if displaying message text for message classes <code>msgClass</code>. If used without an argument, lists all classes.</td>
</tr>
</tbody>
</table>

Table 3-1 Command line options
Using Model Debugger

Table 3-1 Command line options (continued)

<table>
<thead>
<tr>
<th>Short</th>
<th>Long option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e</td>
<td>--env-connect</td>
<td>Connect to remote CADI simulation using the following environment variables:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CADI_CLIENTPORT_TCP – port number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CADI_INSTANCEID – component instance name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CADI_APPLICATIONFILENAME – application file name</td>
</tr>
<tr>
<td>-f</td>
<td>--config-file filename</td>
<td>Use model parameters from the configuration file filename. See Configuration file syntax on page 3-4. See also String syntax on page 3-4.</td>
</tr>
<tr>
<td>-h</td>
<td>--help</td>
<td>Print the available options and exit.</td>
</tr>
<tr>
<td>-i</td>
<td>--instance</td>
<td>Specify the instance.</td>
</tr>
<tr>
<td>-L</td>
<td>--cadi-log</td>
<td>Log all CADI calls into an XML logfile CADILog-nnnn.xml, where nnnn is a set of four digits. The logfile is created in the same directory as the model.</td>
</tr>
<tr>
<td>-l</td>
<td>--list-connections</td>
<td>List possible connections to remote CADI simulations on the local machine and exit afterwards. Each simulation is given a unique simulation ID.</td>
</tr>
<tr>
<td>-m</td>
<td>--model filename</td>
<td>Load the model library in the file named filename. See String syntax on page 3-4.</td>
</tr>
<tr>
<td>-n</td>
<td>--no-params-dialog</td>
<td>Do not display the parameter configuration dialog at startup.</td>
</tr>
<tr>
<td>-O</td>
<td>--sdtout-to-stdout</td>
<td>Redirect standard output to the default destination.</td>
</tr>
<tr>
<td>-p</td>
<td>--project filename</td>
<td>Load the project file filename. See String syntax on page 3-4.</td>
</tr>
<tr>
<td>-q</td>
<td>--quiet</td>
<td>Suppress all Model Debugger and Model Shell output.</td>
</tr>
<tr>
<td>-s</td>
<td>--script filename</td>
<td>Execute the commands from the script named filename. See String syntax on page 3-4.</td>
</tr>
<tr>
<td>-V</td>
<td>--verbose</td>
<td>Equivalent to --enable-verbose &quot;MaxView&quot;.</td>
</tr>
<tr>
<td>-v</td>
<td>--version</td>
<td>Print the tool version and exit.</td>
</tr>
</tbody>
</table>
String syntax

Filenames, and similar strings, included when starting Model Debugger from the command line must be within double quotes if there is white space in the string. For example:

```
modeldebugger -a "my application file.axf"
```

There is, however, no requirement to use quotes if your parameter is a single word with no spaces. This means both of the following forms are valid:

```
modeldebugger --script myscript.txt
modeldebugger --script "myscript.txt"
```

Configuration file syntax

You can configure a model that you start from the command line with Model Debugger by including a reference to an optional plain text configuration file. Each line of the configuration file must contain the name of the component instance, the parameter to be modified, and its value. The following format must be used:

```
instance.parameter=value
```

The `instance` can be a hierarchical path, with each level separated by a dot “.” character. If `value` is a string, additional formatting rules might apply. See String syntax.

You can include comment lines in your configuration file. Such lines begin with a # character. Boolean values can be set using either true/false or 1/0. A sample configuration file including a variety of syntax examples looks like this:

```
# Disable semihosting using true/false syntax
```
coretile.core.semihosting-enable=false
#
# Enable VFP at reset using 1/0 syntax
coretile.core.vfp-enable_at_reset=1
#
# Set the baud rate for UART 0
baseboard.uart_0.baud_rate=0x4800

Running Model Debugger without a GUI

Running a simulation using Model Debugger can be scripted with the MxScript language. A scripted simulation typically does not require control of the target system beyond the provided script. Model Debugger can therefore be run without a Graphical User Interface (GUI). This mode is triggered by the command line option --nogui.

——— Note ————
A simulation platform hosted by Model Debugger in non-GUI mode does not require a script.

You can therefore also run Model Debugger in non-GUI mode without any scripted interaction.

——— ————

To limit the duration of a simulation in non-GUI mode, specify the amount of seconds or system cycles using the command line options:

• --timelimit time_in_seconds
• --cyclelimit number_of_system_cycles.

The timelimit and cyclelimit options are only enabled in --nogui mode.

3.1.2 Launching from System Canvas

To launch Model Debugger from System Canvas for Fast Models click the Debug button on the System Canvas toolbar. See Figure 3-1 on page 3-6.

——— Note ————
If no project is loaded, the Debug button is greyed out.

If you have loaded a project but not built it, System Canvas warns that the target does not exist. You can continue to start Model Debugger but no model is loaded.

——— ————
Figure 3-1 Launching Model Debugger using the System Canvas toolbar

Alternatively, in the main System Canvas menu click Project → Launch in Model Debugger. Model Debugger supports debugging of multiple components.

Using the Configure Model Parameters dialog

At startup, Model Debugger checks the available components and opens a dialog similar to the one shown in Figure 3-2.

--- Note ---

The exact contents and titles of the panes might differ significantly from those shown in the example as they are entirely dependent upon the model you are using.

Figure 3-2 System Canvas Configure Model Parameters dialog
The Configure Model Parameters dialog shown in Figure 3-2 on page 3-6 has the following sections:

**Parameter category**

This pane contains a hierarchical list of the component parameters by category. Click the + symbol to expand the view or the - symbol to collapse the view.

**Parameter setting**

The parameter values are displayed in the right-hand pane.

Use the box in the lower left corner of the window to toggle between hexadecimal or decimal views.

--- Note ---

The name of this pane might be different depending on the model you are using, but its purpose and usage is the same in all cases.

You can also view all of the configuration parameters as a single list as shown in Figure 3-3 by clicking the **List View** tab. The **Tree View** is similar except that it shows the same parameters in a grouped hierarchy.

![Figure 3-3 System Canvas Configure Model Parameters dialog, list view](image-url)
Set the parameters for the model and click **OK** to close the Configure Model Parameters dialog.

**Using the Select Targets dialog**

After closing the Configure Model Parameters dialog, the Select Targets dialog in Figure 3-4 opens to enable you to select the debuggable components in the model.

![Select Targets dialog](image)

**Figure 3-4 Select Targets dialog**

1. Click the box next to the components that to load.
2. The **Application file** column displays applications to load for the cores. If the correct application is not selected, click in the field and enter the name of the source file.
3. If the application name in the list is the same as the application that was already loaded, the debug information is automatically loaded to the debugger.
4. Click **OK** to close the dialog.
5. One or more instances of Model Debugger are created, depending on how many targets you selected to load.

   If the application is loaded and the source code can be found from the debug information in the specified file, Model Debugger displays the code in the source window.

   If the debug information cannot be found because, for example, the **Application file** field is empty, use the Load Application dialog to specify the location for the source code.
Using the Load Application dialog

If the application and source code is not loaded automatically, select Load Application from the Model Debugger File menu to manually locate and load the code.

Reset the component after loading the source.

![Load Application dialog](image)

Figure 3-5 Load Application dialog

3.1.3 Launching Model Debugger separately

Any model that has a CADI interface can be debugged with Model Debugger. You can connect to remote System Generator or SystemC model simulations.

1. Start your model, for example using Model Shell with a CADI server enabled.
2. Launch Model Debugger.
3. Select Connect to Model from the Model Debugger File menu to display the Connect Remote dialog shown in Figure 3-6 on page 3-10. Select and then click Connect to connect to the simulation.
Using Model Debugger

4. Select a target, for example, the processor Core as shown in Figure 3-7, and click **Connect** to connect to the specific component instance in the simulation. If you select more than one instance, one Model Debugger window is opened for each component. Click **OK** to close the dialog.

5. If the source code is not automatically displayed, select **Load Application Code** from the **File** menu. Select the application image from the Load Application dialog and click **Open**.

   **Note**
   If the application is loaded and the source code can be found from the debug information in the application file, Model Debugger displays the code in the source window.
3.2 Model Debugger application windows

The Model Debugger layout consists of the following windows and graphical elements:

- main menu
- tool bar
- workspace with dock windows.

Model Debugger provides a workspace that can contain any of the following view types that are supported by the debugger:

- source code
- disassembly
- call stack
- thread
- register
- memory
- pipeline overview
- pipeline table
- global variables
- local variables
- output
- watch.

Note

The default layout does not contain the following windows:

- thread
- pipe
- global variables
- watch.

The workspace layout can be customized by opening additional views, closing unwanted views, or specifying options in the Preferences dialog. See Preferences dialog on page 3-60.

Note

All views can be moved or resized. Project files enable saving and restoring the customized layout. The files can give each processor target type, and even each target instance, a unique appearance. See Saving the window layout on page 3-23.

The default layout for Model Debugger is displayed in Figure 3-8 on page 3-12.
3.2.1 Main toolbar

The main toolbar, shown in Figure 3-9, provides buttons for frequently used functions. If the functionality is not available in the current context, the buttons is grayed out.

**Open**

Click to open a model library and application file. When the button is clicked:

1. If a model library is not already open, a dialog is displayed to enable you to select a model library to load. Select the model library and click **OK**.
2. If an application is not already open, a dialog opens to enable you to select the application file to load into the target. Select the application file and click **OK**.
3. If a model library and application are already open, a dialog is displayed to select the source file for the application.
   Select the source file and click OK.

--- Note ---
If you are using a Symmetric Multi-Processor (SMP) model with more than one core, such as one based on the Cortex™-A9, then Model Debugger only loads one image that is run on all cores. All Model Debuggers attached to the SMP model load the debug information for that image. This is called SMP awareness.

In certain circumstances SMP awareness can be switched on or off by using the Model Debugger Preferences dialog. See Preferences dialog on page 3-60.

---

Bkpts  Click to open the breakpoint manager

Run  Click this button to run the simulation until a breakpoint is hit or some exception occurs. Encountering a simulation halt is an example of an exception that stops simulation.

Pause/Cont  Click to pause or continue the current high-level simulation step command. An example would be a source-level step. The button text and icon changes depending on whether the simulation is running (Pause) or stopped (Cont).

High level simulation control commands can be interrupted by breakpoints before completion. These commands can be completed by clicking the Cont button.

Stop  Click to stop the execution of the model being debugged.

Step  Click to cause a source-level step to execute until the simulation reaches a different source line.

Over  Click to cause source-level steps to execute the simulation and step over any function calls.

Out  Click to cause source-level steps to execute control command until the current function is exited.

i Step  Click to advance the simulation by executing one source-level instruction.

i Over  Click to advance the simulation by one source-level instruction without following any call instructions.
This command is not supported by all model targets.

Note

i Out
Click to advance the simulation until a return instruction is executed.

Note
This command is not supported by all model targets.

i Step n
Click to advance the simulation by executing the number of source-level instructions specified in the <-n-> control.

Cycle
Click to advance the simulation by a single cycle.

Cycle n
Click to advance the simulation by the number of cycles specified in the edit box. The default is 1000 cycles.

<-n ->
Enter the number of cycles to step if the Cycle n or Back n buttons are clicked. The default is 1000 cycles.

If the i Step n button is clicked, this control indicates the number of instructions to step.

Back n
Click to step the simulation backwards by the number of cycles specified in the edit box. The default is 1000 cycles.

Note
This command is not supported by all model targets.

Back
Click to step the simulation backwards by one cycle.

Note
This command is not supported by all model targets.

Reset
Click to cause a reset of the target model.

Main
Click to cause a reset of the target model. The model runs until the main() function of the application source code is reached.

Note
This command is only available if a function main() can be found in the debug information of the application file.
3.2.2 Menu bar

The main menu bar provides access to most Model Debugger functions and commands.

**File menu**

The File menu has the following options:

**Open Source ...**

Opens the source code for the application.

**Source File Manager ...**

Displays the Source File Manager dialog.

**Load Application Code ...**

Loads application code to the model.

**Load Application Code (Debug info only) ...**

Loads debug information only.

**Load Model ...**

Loads a model.

**Connect to Model ...**

Displays the Connect to Target dialog to connect to a model file.

**Close Model**

Closes the currently open model. If Model Debugger is connected to a CADI server, Model Shell for example, the connection is closed but the simulation continues to run.

**Open Session ...**

Opens a previously saved session.

**Save Session**

Saves the current debug session.

**Save Session As**

Saves the current debug session to a new location and name.

**Preferences**

Displays the Preferences dialog to enable you to modify the user preferences.
Recently Opened Models

Displays a list of the most recently opened model files. Click on a list entry to open the file. By default, the last sixteen files are displayed in the list. The number of files to display can be set in the Preferences dialog.

To remove a file from the list, move the mouse cursor over the file name and press the Delete key or right click and select Remove from list from the context menu.

Recently Opened Applications

Displays a list of the most recently opened applications. Click on a list entry to open the application. By default, the last sixteen applications are displayed in the list. The number of applications to display can be set in the Preferences dialog.

To remove an application from the list, move the mouse cursor over the application name and press the Delete key or right click and select Remove from list from the context menu.

Recently Opened Sessions

Displays a list of the most recently opened sessions. Click on a list entry to open the session. By default, the last sixteen sessions are displayed in the list. The number of sessions to display can be set in the Preferences dialog.

To remove a session from the list, move the mouse cursor over the session name and press the Delete key or right click and select Remove from list from the context menu.

Exit

Ends Model Debugger. If you have modified files or sessions, a dialog prompts you to save your changes.

Search menu

The Search menu has the following options:

Find ...

Opens a dialog that enables searching for a string in a currently active window.

Find Next

Repeats the last defined search to find the next occurrence.

Find Previous

Repeats the last defined search, but the search direction is backwards in the document.
Control menu

The Control menu has the following options:

**Restart Simulation ...**

This option restarts the simulation.

**Reset**

Click to cause a reset of the target model.

**Goto Main**

Cause a reset of the target model. The model runs until the `main()` function of the application source code is reached.

--- Note ---

This command is only available if a function `main()` can be found in the debug information of the application file.

**Run**

Run the simulation until a breakpoint is hit or some exception occurs. An example would be simulation halt.

**Pause/Continue Source Step**

Pause or continue the current high-level simulation step command. An example would be a source-level step.

**Source Step Over**

Cause a source-level step to execute until the simulation reaches a different source line.

**Source Step Out**

Cause source-level steps to execute control command until the current function is exited.

**Instruction Step**

Advance the simulation by executing one source-level instruction.

**Instruction Step Over**

Advance the simulation by one source-level instruction without following any call instructions.

--- Note ---

This command is not supported by all model targets.

**Instruction Step Out**

Advance the simulation until a return instruction is executed.
Note

This command is not supported by all model targets.

---

**Instruction Step n**

Advance the simulation by the number of instructions specified in the `<n>` edit box. The default is 1000 cycles.

**Cycle Step**

Advance the simulation by a single cycle.

**Cycle Step n**

Advance the simulation by the number of cycles specified in the edit box. The default is 1000 cycles.

**Enable/Disable Step Back**

Enable or disable stepping back by cycles.

Note

This command is not supported by all model targets.

---

**Back**

Step the simulation backwards by one cycle.

Note

This command is not supported by all model targets.

---

**Back n**

Step the simulation backwards by the number of cycles specified in the edit box. The default is 1000 cycles.

Note

This command is not supported by all model targets.

---

**Debug menu**

The **Debug** menu has the following options:

**Display Messages**

Display debug messages.

**Clear Log**

Clear the log of debug messages.

**Clear Model Output**

Clears all output messages from the model.
Clear Output Summary

Clear the summary output messages.

Breakpoint Manager ...

Display the Breakpoint Manager dialog.

Profiling Manager ...

Display the Profiling Manager dialog.

View Profiling ...

Display the Profile Information dialog.

Save Model State ...

Save the current model state. If reloaded, simulation continues from the point where the model state was saved.

Restore Model State ...

Reload a previously saved model state.

Load Debug Info for Module

Load debug information for the module.

Set Parameters

Set parameter values for the model.

Select Targets

Select the execution target within the model.

Layout menu

The Layout menu has the following options:

Layout Control Window

Display the Window to set layout options such as tiling.

Load Layout ...

Load a previously saved window layout.

Save Layout ...

Save the current layout. Model state is not saved.

Load Recent Layout

Use a recently used window layout.
Restore Default Layout

Restore the window layout to the defaults. This option is useful if the layout has become disorganized.

Window menu

The Window menu has the following options:

New View Display a new debug view.
Hide Hide an existing debug view.
Show Display view that was most recently hidden.
Show All Displays all previously hidden views.
Close Close the window that currently has focus.

Arrange Horizontally

Tile all view windows horizontally.

Arrange Vertically

Tile all view windows vertically.

Move Move a view to the new position specified on the submenu.

Docked Views

Dock or undock the views listed on the submenu.

Help menu

The Help menu has the following options:

Help ... Opens this book in Adobe Acrobat Reader.
About ... Displays the standard About dialog box displaying version and license information.

About Model ...

Opens the text file that contains the release notes.

3.2.3 Dock windows

Model Debugger provides dock windows that can be docked inside the main workspace or floated as a top level window. To toggle between the docked and floating state, double-click on the dock window handle or the title bar of the floating window.
3.2.4 Moving or copying views

Debug views can be moved within the same dock window or copied by dragging and dropping into another dock window. To start a drag-and-drop operation, left-click the debug view and, while holding the mouse button down, press the F9 key.

Figure 3-10 shows the window appearance while you are moving the Memory window. The target location for the example in Figure 3-10 is indicated by a gray box on the left edge near the bottom of the Model Debugger window. Releasing the mouse button drops the window into the gray box.

Press the Ctrl + F9 to copy the window. This effectively duplicates the existing view. The location for the duplicate view in Figure 3-11 on page 3-22, for the Local Variables window, is indicated by a gray box near the centre of the Model Debugger window. Releasing the mouse button creates the duplicate window in the target location.
**Figure 3-11 Duplicating a register view**

_____ Note _______

The windows might be difficult to place into the required position. To force the window to dock to a particular location, select the window handle and right-click to display the context menu:

- select **Dock Right** to force the selected window to the right edge
- select **Dock Left** to force the selected window to the right edge
- select **Dock Top** to force the selected window to the top edge
- select **Dock Bottom** to force the selected window to the bottom edge

It might be necessary to combine different move operations to force the window to the required location.
3.2.5 Saving the window layout

If you use different debug windows and views for different models, you can save and later reload layouts to simplify reorganizing the views.

The Layout menu has the following entries:

**Layout Control Window**

Displays the window shown in Figure 3-12. Click on an entry to change focus to the selected window.

![Figure 3-12 Layout Control window](image)

Right-click to display a context menu for moving or duplicating windows.

![Figure 3-13 Layout Control context menu](image)

--- Note ---

You can also use drag-and-drop within the Layout Control window to change the location of the windows.

---
Load Layout

Load a previously saved layout file. The window positions match the window configuration present when the layout was saved.

![Load Layout dialog](image)

**Figure 3-14 Load Layout dialog**

Save Layout

Save the current window arrangement to a layout file.

Load Recent Layout

Load the last saved layout. If you have modified the current layout, a prompt asks whether to save the current layout.

Restore Default Layout

Use the default layout.

### 3.2.6 New debug views

Open a new window by either:

- selecting **New View** from the **Window** menu and selecting the desired type of debug view.
- clicking on the **View** icon located at the right of the menu bar and selecting a view from the list. See Figure 3-15 on page 3-25.
3.2.7 Closing windows and views

You can close a dock window, and all views in the window, by clicking the close button in the dock handle or title bar. This closes all views contained in the window.

Views can be closed individually by clicking on their close icon. See Figure 3-16.

Figure 3-15 Icons for selecting a new debug view

Figure 3-16 Closing windows or individual debug views
3.2.8 Output window

This window displays messages from Model Debugger and from the targets being debugging. The window has the following tabs:

- **Log** contains debugger messages, such as errors and warnings
- **StdIO** contains output from the target model
- **All** is an interleaved view for both the Log and StdIO categories.

The Command text box is located next at the bottom of the Output window.

To execute a command, enter the command text in the text field and click the **cmd>** button. The available commands are described in a separate book. See the *MxScript Reference Manual*. 
3.3  Debug views for source code and disassembly

The Source code and Disassembly views share a common window. Each view consists of:

- a title bar with controls for selecting a target line or switching between views
- the actual code browser for source or disassembly
- columns for line number or address.

The function of the columns and title bar controls is specific to each view.

3.3.1  Source view

The Source view contains two columns with a gray background on the left that contain the line number and bullets that represent executable code locations. The right side of the view contains your source code. See Figure 3-18 on page 3-28 for an example.

The button with the green arrow shown in Figure 3-17 scrolls the code browser to the location of the statement or instruction that is to be executed next. You can find this button at the top left of the Source view window.

![Figure 3-17 Arrow button for scrolling code](image)

Click on the left-most column in the Source code view to highlight the corresponding addresses in the disassembly view. The highlighting reveals the instructions the source statement maps to.

--- Note ---

Highlighting is only available for source lines with a bullet. The bullet indicates that the line is executable.

---

Double-click on a bullet to set a breakpoint on the source line. A red disk is displayed next to the line to indicate that a breakpoint has been set, as shown in Figure 3-18 on page 3-28.
The Source code view title bar has additional controls for:

- selecting a target line in the source using the **Line:** entry box
- selecting a source file that has already been loaded using the **File:** drop down list
- opening the Debug Source Files dialog.

**Context menu for Source view**

Right-click in the Source view to display the context menu. The menu has the following options:

**Insert Breakpoint**

Insert a breakpoint at the selected location.

**Enable Breakpoint**

Enable the breakpoint at the selected location.

**Breakpoint Properties**

If a breakpoint is present on the selected instruction, selecting this option displays the Breakpoint properties dialog.

**Run to here**  
Run to the selected instruction

**Word wrap**

Wrap text to fit inside window

**File properties**

Display the filename and path for the currently displayed file.
Debug Source Files dialog

The Debug Source Files dialog, shown in Figure 3-19, lets you locate source files required for debugging an application. To open the dialog, click on the icon in the upper right corner of the Source view.

— Note —--------

Pathnames are displayed with slash (/) characters, even on Windows. This does not affect operation.

—-----------------

Figure 3-19 Debug Source Files dialog

The tabs switch between two different views that list the properties for the source file:

**Filename**

This column contains a list of files referred to by the debugged application. This column is not shown in **Hierarchy** view.

**Debug pathname**

This column shows the path for the respective file taken from the debug information of the application. This path might be invalid as it refers to the original source file at compilation time. The debug pathname can be absolute or relative to the executable.

**Actual pathname**

This column contains the path Model Debugger actually uses to locate the file. If the last column is empty or contains an invalid path, the path can be set by double-clicking a row or selecting a row and clicking **Open File**. The File Open dialog displays to enable selecting the source file. After selecting the file, the file is opened in the debugger.
Click on **Find File** to display the Find source file dialog shown in Figure 3-20 and navigate to the directory containing the source.

![Figure 3-20 Find Source File dialog](image)

Click on **Properties** to display the File Properties dialog for the selected file as shown in Figure 3-21. You can also use the **Find File** button on the File Properties dialog to locate the file.

![Figure 3-21 Source File Properties dialog](image)

Model Debugger has an automatic mechanism to add replacement paths that are invoked every time you are prompted to find a source file. If the source file is found, an automatic *source path replacement* is calculated.
This path might not always be correct, and there are situations where you must manually edit source path replacements because the automatic path is wrong for the specific context. You might, for example, have a header file whose name is common between two different compilers, and Model Debugger chooses the wrong one.

Click on **Source Paths...** to open the Source Path Replacements dialog as shown in Figure 3-22. Use this dialog to change the path, or priority of the paths, to the source files for the application.

--- Note ---

The source path replacements are stored in the Model Debugger session file and not with user preferences.

![Figure 3-22 Source Path Replacement dialog](image)
Existing source file replacements are displayed in the top part of the Source Path Replacement dialog. You can remove or reorder paths by highlighting an entry and clicking one of the following buttons:

**Move Up**  
Move the selected path up one position in the list.

**Move Down**  
Move the selected path down one position in the list.

**Remove Entry**  
Delete the selected path from the list.

**Debug Path** and **Actual Path** have the same meaning as in the Debug Source Files dialog shown in Figure 3-19 on page 3-29.

In the lower part of the Source Path Replacement dialog, you can add new source paths or modify existing ones. The additional features are:

**Debug info paths**  
Provides a tree view that simplifies navigation through the debug paths found in the debug information of the source file.

**Browse**  
Click this button to select an path with a browser rather than typing in the actual path directly.

**Apply Changes**  
Modify the selected entry using the changes entered.

**Insert as New Entry**  
Adds the new path to the source path replacement list.

**Searching in source files**

You can search for text in the active window by using the Find dialog. The active window is surrounded by a thin black frame. Click **Find** on the **Search** menu to open the Find dialog shown in Figure 3-23 on page 3-33:
Type the text to find in the box and click the **Find Next** or **Find Previous** buttons to search upwards or downwards from the current cursor position. You can re-use previous search terms by clicking on the drop-down arrow on the right of the text entry box.

The dialog is modeless, so you can change views without closing the dialog. The mode is updated automatically.

### 3.3.2 Disassembly view

The Disassembly view provides three columns for the address, opcode, and disassembly string as shown in Figure 3-24.

Move the cursor over a disassembly line to display the whole disassembly in a help bubble. This is particularly useful if the complete disassembly string does not fit horizontally into the view.

![Figure 3-24 Disassembly view](image)

The Disassembly view title bar has the following controls:
- **Address:** to enter a start address to display the code from
Memory space:

to select Secure (TrustZone®) or Normal memory space, if applicable for the processor architecture

Architecture

to select the disassembly mode or instruction sets for the opcodes, such as ARM or Thumb.

Mapping source lines to the disassembly listing

Click on the left-most column in the source view to highlight in blue the corresponding addresses in the disassembly view. The highlighting indicates the disassembly instructions to which the respective source statement maps.

--- Note ---
This action is only possible for source lines with a bullet point.

Figure 3-25 Matching source and disassembly

Context menu for Disassembly view

Right-click one in the Disassembly view to display the context menu. The menu has the following options:

Insert Breakpoint

Insert a breakpoint on the selected location.

Enable Breakpoint

Enable the breakpoint at the selected location.
Breakpoint Properties

If a breakpoint is present on the selected location, selecting this option displays the Breakpoint properties dialog.

Show memory

Display a dialog to select a memory space and update the Memory view to display the memory contents at the address specified corresponding to the instruction location.

Run to here  Step the code until the selected location is reached.

3.3.3 Call Stack

The Call Stack view displays the call history.

To use the Call Stack view, DWARF register mapping must be defined for the architecture and provided in the model.

Note

The loaded application must be an ELF file that contains a .debug_frame section. No other type of debug information is supported for the call stack view. The .debug_frame section must contain valid DWARF debug information that matches the DWARF 2 or DWARF 3 specification. This is provided by the C compiler to inherently describe all necessary information to unwind the stack:

- the stack pointer
- how to retrieve the previous frame pointer
• all registers involved in the unwinding process.

This information cannot be supplied by any other means than the frame section.
3.4 Debug views for registers and memory

This section describes the views related to register or memory contents.

3.4.1 Register views

The Register view displays registers and their values and organizes them into multiple groups. A combo box enables switching between the groups that are predefined by the target model. See Figure 3-27.

![Figure 3-27 Select register group](image)

For each register, a buffered state of the register (previous value) is stored. To view the contents, either:

- use the context menu in the register view and select **Show Previous Values**. The contents are displayed as shown in Figure 3-28:

![Figure 3-28 Register view showing current and previous contents](image)

- place the cursor over the respective register as shown in Figure 3-29 on page 3-38. The buffered state is updated every time the model execution stops.
Context menu for Register view

Right-click one of the registers in the Register view to display the context menu. The menu has the following options:

**Copy**
Copy the contents of the selected register.

**Add to Watch**
Add the selected register to the Watch view.

**Insert Breakpoint**
Insert a breakpoint on the selected register.

**Enable Breakpoint**
Enable the breakpoint at the selected register.

**Breakpoint Properties**
If a breakpoint is present on the selected register, selecting this option displays the Breakpoint properties dialog.

**Edit Value**
Edit the contents for the selected register.

**Select and show memory at nnn**
Display a dialog to select a memory space and update the Memory view to display the memory contents at the address specified by the register contents.

**Show memory at nnn**
Update the Memory view to display the memory contents at the address specified by the register contents.
Format

Choose the number base to use to display the register contents. The options are Default Format, Unsigned Decimal, Signed Decimal, Hexadecimal, Binary, Float, or ASCII.

Show Previous Value

Display the current value and the previous value for the selected register.

Select All

Select all of the registers in the Register view.

3.4.2 Memory

The Memory view displays a range of memory starting from the base address specified in the address field (Addr:). The base address can be entered as a decimal number or, by using the prefix 0x, as hexadecimal numbers. Additional fields are provided for selection of the address space (Space:) and physical memory block (Block:). See Figure 3-30.

![Figure 3-30 Memory view](image)

Context menu for Memory view

Right-click one of the cells the Memory view to display the context menu. The menu has the following options:

Insert Breakpoint

Insert a breakpoint on the selected memory location.

Enable Breakpoint

Enable the breakpoint at the selected memory location.
Breakpoint Properties
If a breakpoint is present on the selected memory location, selecting this option displays the Breakpoint properties dialog.

Edit Value
Edit the contents for the selected memory location.

Select and show memory at nnn
Display a dialog to select a memory space and update the Memory view to display the memory contents at the address specified by the contents of the memory location.

Show memory at nnn
Update the Memory view to display the memory contents at the address specified by the contents of the memory location.

Show disassembly at nnn
Update the disassembly view to display the disassembly contents at the address specified by the contents of the memory location.

Copy
Copy the contents of the selected memory location.

Add to Watch
Add the selected memory location to the Watch view.

Endian
Select the memory model to use to display memory contents. The options are: Default Endian, Little Endian, and Big Endian.

Format
Choose the number base to use to display the memory contents. The options are Default Format, Unsigned Decimal, Signed Decimal, Hexadecimal, Binary, Float, or ASCII.

Fixed column count
Display a fixed number of memory values per row. The number to be displayed is determined by the width of the memory window.

Increment column count
Increment the number of memory values to display per row.

Decrement column count
Decrement the number of memory values to display per row.

Increment current address
Increment the start address used for each memory row.
Decrement current address

Decrement the start address used for each memory row.

Increment MAU per cell

Increase the size of the word, that is, the Minimum Addressable Unit (MAU), to be displayed in each memory cell. This also changes the memory access size. If the chosen access size is not supported, Model Debugger defaults to a size of a single MAU.

Decrement MAU per cell

Decrease the size of the word, meaning MAU, to be displayed in each in each memory cell. This also changes the memory access size. If the chosen access size is not supported, Model Debugger defaults to a size of a single MAU.

Memory Display Options

Display the Memory Display Options dialog to enable setting column count, view format, endian mode, and MAU per cell.

3.4.3 Variables

Variables are displayed in the following windows:

Local Variables window

This window shows all local variables and parameters that are valid for the current PC value. The type and value of the variables are displayed as shown in Figure 3-31 on page 3-42:

• a blue L before the variable name indicates a local variable
• a green leading P indicates a parameter.
Global Variables window

This window shows the global variables with their types and values. They are marked by a green G as shown in Figure 3-32.

Complex values such as structs and arrays or pointers can be expanded by clicking the small cross before the variable name.

—— Note ———

To use the variables windows, the loaded application must be an ELF file that contains .debug_info and .debug_abbrev sections. No other type of debug information is supported for this view. The .debug_info section must contain valid DWARF debug information that matches the DWARF 2 or DWARF 3 specification. The model must provide a PC register to enable locating local variables.

For applications that have more than one compilation unit, the units are only loaded when the PC reaches the respective context.
The loading of these compilation units can be triggered manually by selecting **Load Debug Info for Module** from the **Debug** menu. Right-click on one of the variables windows and select **Load Debug Info for Module**.

The displayed dialog lists all of the compilation units that can be loaded.

**Context menu for Variable view**

Right-click one of the items in the Global or Local Variable view to display the context menu. The menu has the following options:

- **Copy**
  - Copy the contents of the selected variable.

- **Add to Watch**
  - Add the selected variable to the Watch view.

- **Insert Breakpoint**
  - Insert a breakpoint on the selected variable.

- **Enable Breakpoint**
  - Enable the breakpoint at the selected variable.

- **Breakpoint Properties**
  - If a breakpoint is present on the selected variable, selecting this option displays the Breakpoint properties dialog.

- **Edit Value**
  - Edit the contents of the selected variable.

- **Show memory**
  - Display a dialog to select a memory space and update the Memory view to display the memory contents at the address specified by the value of the variable.

- **Show Previous Value**
  - Display the current value and the previous value for the selected variable.

- **Select All**
  - Select all of the variables in the variable view.

- **Load Debug Info for Module**
  - Load debug information for the module that contains the selected variable.
3.5 Debug views for pipelines

Model Debugger provides the following options for viewing the pipeline:
• the Pipeline Overview window
• the Pipeline Table.

Note
Pipeline views are only available if your model supports them. If the pipeline icons are grey, not orange, then you cannot view pipeline information.

3.5.1 Pipeline Overview window

The Pipeline Overview window gives a brief pipeline overview that presents the main details of every pipeline stage. The Pipeline Overview contains the name, program counter, opcode and disassembly for the stages, as shown in Figure 3-33.

![Figure 3-33 Pipeline Overview window](image)

3.5.2 Pipeline Table window

The Pipeline Table shown in Figure 3-34 on page 3-45 gives a very detailed view of the pipeline stages. By default, the table shows views for all pipeline stages. Each of the detailed entries has a name and value field.
Columns and rows can be resized by grabbing the lines between them and dragging them.

The cross in the top left corner of every pipeline stage view is the starting point for drag and drop operations. A view can be copied or moved to an empty table cell. If it is dragged beyond the boundaries of the table, a new row or column is added in the direction of the drag.

Double-click on the first column of a view to set or remove a breakpoint on the field.

Double-click on the second column of a view to open the field for inline edit of the value. You cannot, however, perform an inline edit of an opcode or disassembly.

**Pipeline Table context menu**

Right-click into empty space of the table or in an empty cell to open the context menu shown in Figure 3-36 on page 3-46:
The context menu has the following entries:

**New Pipeline Stage**
Select to create a new stage and add it to the view.

**Reset to default**
Select to use the default layout for the Pipeline Table view.

**Insert Row Above/Insert Row Below**
Select to insert a new row above or below the current cell.

**Insert Column Left/Insert Column Right**
Select to insert a new column to the left or right of the current cell.

**Remove Row/Remove Column**
Select to remove the row or column that includes the current cell.

**Save Pipeline Table Layout**
Select to save the current layout to file.

**Load Pipeline Table Layout**
Select to load a previously saved layout and use that configuration in the Pipeline Table view.
Pipeline Stage Properties dialog

Click the orange icon to the right of the cross to open the Pipeline Stage Properties dialog shown in Figure 3-37. Use this dialog to customize the lists in the Pipeline table. The combobox to the right of the orange icon lists all pipeline stages that are available for the current model. The chosen pipeline stage is displayed in the view underneath. Click on the X, located in the right top corner of each list, to remove the view from the cell.

![Figure 3-37 Pipeline Stage Properties dialog](image)

Choose the fields to be displayed in the pipeline stage list by checking the boxes or clicking the **Show All** or **Hide All** buttons.

The size of the first column can be set in the **Visible characters in the name column** control.

The optional header can be switched on and off by checking the **Show header** check box.

**Context menu for an entry in the Pipeline Table**

Right-click on an item in the Pipeline view lists to open the context menu shown in Figure 3-38 on page 3-48.
The context menu has the following entries:

**Close Pipeline Stage**
Select to close the pipeline stage.

**Copy**
Select to copy the pipeline field. The field can be pasted into the Watch window.

**Add to Watch**
Select to add the pipeline field to the Watch window.

**Remove Breakpoint/Insert Breakpoint**
The text displayed depends on whether the selected field already has a breakpoint:
- If a breakpoint is present, select **Remove Breakpoint** to delete it.
- If a breakpoint is not present, select **Insert Breakpoint** to add a breakpoint.

**Enable Breakpoint/Disable Breakpoint**
If a breakpoint is present:
- select **Enable Breakpoint** to enable it
- select **Disable Breakpoint** to retain the breakpoint, but disable it.
Breakpoint Properties
Select to open a dialog to view and set the details and conditions for a particular breakpoint.

This is the same dialog that is available from the Breakpoint Manager dialog. See Breakpoint Manager dialog on page 3-56.

Edit Value
Select to open the chosen field for inline edit.

Show Memory
Select to mark the memory address in the Memory Window.

Format
Select to open the submenu shown in Figure 3-39. This menu lets you choose the format for number display.

![Figure 3-39 Submenu for display format](image)

Select All
Selects all fields in the list.

Pipeline Stage Properties
Select to open the Pipeline Stage Properties dialog change the contents of the Pipeline Stage view. See Figure 3-37 on page 3-47.

Pipeline Table
Select to display a submenu for the Pipeline Table. See Figure 3-36 on page 3-46.
3.6 Watch window and Expression Evaluator

The Expression Evaluator is located in the Watch window. See Figure 3-40. To display the window, select Window → New View → New Watch Window.

![Watch window](image)

Figure 3-40 Watch window

There are two types of entry in the Watch window:

**System variables**

Entries in this group are marked with small icons to the left of their name to indicate their origin. They can be manipulated in the Watch window in the same was as in their original view. Items in this category include:

- registers
- memory locations
- pipeline fields
- variables from source code.

**Expressions for evaluation**

These items do not have an associated icon because they are not duplicates of an item in a different view. They cannot have breakpoints set and their value cannot be changed. The expression itself can, however, be edited by text in the Name column.

Double-click in the left column of an existing entry to add a breakpoint for that variable.

Double-click in the right column of an existing entry to edit the contents.

Double-click on the last entry in the left column to enter a new expression. Press the Enter key to perform the evaluation.

The following rules apply to the names of the resources in the target:

- Registers must be entered in the form:
  
  `$registerGroup.registerName`

  If the register name is unique for the whole target the following shorthand notation can be used:
$registerName

- Pipeline stage fields must be entered in the form:
  @pipelineStage.fieldName

- Memory locations must be entered in the form:
  memorySpace:address

  The content of a memory location is queried with the following expression:
  *memorySpace:address

  The delivered pointer can be type cast into any desired type:
  (typeName*)memorySpace:address

The variables of the software running on top of the target processor can be entered using an expression as they appear in the software. They do not require a prefix or quotes. Access components of structs or unions with the '.' or '->' operator according to the C syntax.

Numeric values can be entered in the following formats:

**Integer values**

Integer values can have binary, octal, decimal or hexadecimal representation. The prefix indicates the representation format:

- binary numbers have a leading 0b
- octal numbers a leading 0
- hexadecimal numbers have a leading 0x
- literals with no prefix are interpreted as decimals.

**Floating-point values**

Floating point values can have decimal and scientific representation.

Floating point values can be entered in two different formats: in decimal representation (123.456) or in scientific representation with positive or negative exponent (1.23456e2).
Resources, variables and literals in the target can be combined with operators to form complex expressions. The expression evaluator has the same operands as the C language and has the same precedence and associativity of operators. See Table 3-2. Inside the complex expression, the resources of the target can be used if an integer value would be sufficient in a regular C expression.

### Table 3-2 Operator precedence

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operators</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[ ] -&gt; .</td>
<td>left to right</td>
</tr>
<tr>
<td>2</td>
<td>! ~ + - * &amp; (unary) (cast) sizeof</td>
<td>right to left</td>
</tr>
<tr>
<td>3</td>
<td>* (binary) / %</td>
<td>left to right</td>
</tr>
<tr>
<td>4</td>
<td>+ - (binary)</td>
<td>left to right</td>
</tr>
<tr>
<td>5</td>
<td>&lt;&lt; &gt;&gt;</td>
<td>left to right</td>
</tr>
<tr>
<td>6</td>
<td>&lt; &lt;= &gt; =&gt;</td>
<td>left to right</td>
</tr>
<tr>
<td>7</td>
<td>== !=</td>
<td>left to right</td>
</tr>
<tr>
<td>8</td>
<td>&amp; (binary)</td>
<td>left to right</td>
</tr>
<tr>
<td>9</td>
<td>^</td>
<td>left to right</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>&amp;&amp;</td>
<td>left to right</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>?:</td>
<td>right to left</td>
</tr>
</tbody>
</table>

#### 3.6.1 Context menu for Watch Window

Right-click one of the values in the Watch Window to display the context menu. The menu has the following options:

- **Paste**: Insert a copied memory, register, or variable into the Watch Window.
- **Copy**: Copy an item and its value from the Watch Window.
- **Insert Breakpoint**: Insert a breakpoint on the selected watched item.
Enable Breakpoint

Enable the breakpoint at the selected watched item.

Breakpoint Properties

If a breakpoint is present on the selected watched item, selecting this option displays the Breakpoint properties dialog.

Edit Value

Edit the contents for the selected watched item.

Select and show memory at nnn

Display a dialog to select a memory space and update the Memory view to display the memory contents at the address specified by the contents of the watched item.

Show memory at nnn

Update the Memory view to display the memory contents at the address specified by the contents of the watched item.

Format

Choose the number base to use to display the watched item. The options are Default Format, Unsigned Decimal, Signed Decimal, Hexadecimal, Binary, Float, or ASCII.

Increment number of bytes

Increment the number of memory addresses to display.

Decrement number of bytes

Decrement the number of memory addresses to display.

Select all

Select all of the watched items.
3.7 Setting breakpoints in the debug views

Breakpoints can be set in the following debug views:

Source code view

The second column contains small bullets for each source line where breakpoints can be set. Double click on a bullet to set a breakpoint.

Disassembly view

Double-click on any column to set a breakpoint on that line.

Register view

Double-click on the first column, the register name column, to set breakpoints. An example is shown in Figure 3-43 on page 3-55.
Memory view

Select Insert Breakpoint from the context menu to set breakpoints. It is not possible to set a memory breakpoint by double-clicking on an address.

Local variables view

Double-click on items in the first column to set breakpoints.

Note
To use this view, the loaded application must be an ELF file that contains .debug_info and .debug_abbrev sections.

Global variables view

Double-click on items in the first column to set breakpoints.

Call stack view

Double-click on items in the first column to set breakpoints.

Note
To use this view, DWARF register mapping must be defined for the architecture and provided in the model.

The loaded application must be an ELF file that contains a .debug_frame section.

Pipeline Table

Double-click on the name in the first column to set breakpoints. An example is shown in Figure 3-44 on page 3-56.
Watch view  If an item has been copied from another view into the Watch view, breakpoints can be set in either the original view or the Watch view.

3.7.1  Conditional breakpoints

Conditional breakpoints are supported for some breakpoint objects. Conditional breakpoint are created by:

1. setting an unconditional breakpoint
2. using the Breakpoint Properties dialog to set the conditions for the breakpoint. See Breakpoint Properties dialog on page 3-57.

3.7.2  Removing and disabling breakpoints

You can quickly remove a breakpoint by double clicking on it. To inactivate a breakpoint without removing it, disable the breakpoint by right-clicking on the breakpoint and selecting Disable breakpoint in the resulting context menu. A disabled breakpoint is shown as a gray, rather than red, circle symbol. Other breakpoint dialogs and menus also allow you to configure your breakpoints.

3.7.3  Breakpoint Manager dialog

All breakpoints can be controlled and maintained through the Breakpoint Manager shown in Figure 3-45 on page 3-57. This dialog lists all breakpoints and provides the breakpoint target location, condition, and target details.

Breakpoints that are hit are highlighted in the breakpoint list with an orange background. The breakpoint is also highlighted in the original view for the item.

In the breakpoint list, select an item to:

• enable, disable or remove breakpoints
• locate the breakpoint target location in the respective debug view
• modify breakpoint conditions using the Properties button.
3.7.4 Breakpoint Properties dialog

The Breakpoint Properties dialog can be obtained by right clicking on a breakpoint and then selecting Properties from the resulting context menu, as well as from the Breakpoint Manager. Use the Breakpoint Properties dialog, shown in Figure 3-46 on page 3-58, to select the breakpoint criteria:

**Ignore count**

Enter the number of occurrences to ignore before triggering the breakpoint. Enter 0 to trigger a breakpoint for every occurrence.

**Enable breakpoint**

Check this box to enable the breakpoint. If unchecked, the breakpoint location and type is stored, but occurrences do not trigger a breakpoint.

**Resource**

Select the condition that results in a breakpoint being triggered. Conditional breakpoints are not supported for some types of breakpoint object.

**Value**

If the Resource type is not breaks unconditional, select the comparison value that will trigger the breakpoint.

**Trigger Type**

Select whether a Read, Write, or Modify operation triggers the breakpoint. These check boxes are not enabled for some types of breakpoint object.

**Hexadecimal value display**

Check to display the contents of the Value field in hexadecimal format. If unchecked, decimal format is used.
Figure 3-46 Breakpoint Properties dialog
3.8 Model Debugger sessions

Model Debugger session files enable saving and restoring debugging sessions and provide a convenient way to specify the session parameters. Session files have the extension \*.mvs.

Note

The session files are only available for directly loaded models. They cannot be used for connections to Model Shell or SystemC simulations.

The information that can be saved and restored includes:

• debugger main window geometry
• layout of all debug views
• target model being loaded
• application file
• breakpoints.

Session files also enable configuring the individual layout of debugger windows for multi-core systems. This might be useful, for example, if the project is used with SoC Designer.

To save a session, select Save Session, or Save Session As, from the File menu.

To load a session and restore the original model connection and window layout, select Load Session from the File menu.
3.9 Preferences dialog

Use the Preferences dialog to configure the behavior of Model Debugger.

![Preferences dialog](image)

Figure 3-47 Preferences dialog

Select an entry in the list on the left side of the dialog to display the options for that category:

**Appearance**

Check the boxes to select the following options:

- **Show tool tips** enables display of pop-up help for a control
- **Display toolbar text labels** displays text below the icon
- **Word wrap in source window** wrap long lines to fit the window
- **Show splash screen on startup** displays the information screen
- **Reload recent layout on startup** keeps your last used layout.

Use the controls in the **Recent files and directories** to control how many previously used files and directories are displayed.

**Fonts**

Use this view to specify the fonts for each of the windows.

Check the **Fonts depend on SDISPLAY variable** to have the variable control the fonts.
Suppressed Messages

Lists the suppressed messages and enables you to specify an action for each message.

Verbose Messages

Turn on or off verbose message setting for the message IDs. Click Selective to turn on or off individual messages.

Other settings

Check the boxes to select the following options:

- Load all Compilation Units at Startup to load all required files.
- Show Parameter Dialog at Startup to display the dialog to configure model parameters.
- Show Target dialog at Startup to display dialog that normally appears when a model is loaded. If unchecked, Model Debugger automatically connects to targets that have the executes_software flag set.
- Enable SMP Application Loading to have ModelDebugger load the application only once into memory and load only debug info for all cores. The PC is set to the value of the first core in all cores.

After displaying the dialog and modifying preferences:

- Click Apply to apply the settings and keep the dialog open
- Click OK to apply the settings and close the dialog
- Click Close to close the dialog. Any changes to the settings that have not been activated by pressing Apply are lost.
Appendix A
Shortcuts

This appendix describes shortcuts available in Model Debugger. It contains the following section:

•  *Keyboard shortcuts* on page A-2.
A.1 Keyboard shortcuts

Table A-1 lists the keyboard shortcuts.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ESC</td>
<td>Close the current dialog.</td>
</tr>
<tr>
<td></td>
<td>F1</td>
<td>Help.</td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>Find next.</td>
</tr>
<tr>
<td>Shift</td>
<td>F3</td>
<td>Find previous.</td>
</tr>
<tr>
<td></td>
<td>F5</td>
<td>Run target.</td>
</tr>
<tr>
<td>Shift</td>
<td>F5</td>
<td>Stop target.</td>
</tr>
<tr>
<td></td>
<td>F10</td>
<td>Source-level step over.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>F10</td>
<td>Source-level step out. This leaves the current function.</td>
</tr>
<tr>
<td>F11</td>
<td></td>
<td>Source-level step into.</td>
</tr>
<tr>
<td>Shift</td>
<td>F11</td>
<td>Instruction-level step.</td>
</tr>
<tr>
<td>Shift</td>
<td>F10</td>
<td>Instruction-level step over.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>F11</td>
<td>Instruction-level step out.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>F11</td>
<td>Cycle step.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>F11</td>
<td>Cycle step N.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>B</td>
<td>Open the breakpoint manager.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>C</td>
<td>Connect to model.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>D</td>
<td>Load debug information from application code.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>F</td>
<td>Search (find) operation.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>L</td>
<td>Load model library.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>M</td>
<td>Go to function main().</td>
</tr>
<tr>
<td>Ctrl</td>
<td>O</td>
<td>Multi-functional open.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If no target is loaded, a dialog is displayed to select the model library and application code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If a target is loaded, the source file is opened.</td>
</tr>
</tbody>
</table>
### Table A-1 Keyboard shortcuts (continued)

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + Shift</td>
<td>O</td>
<td>Load application code.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>P</td>
<td>Open the Model Parameter dialog.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>P</td>
<td>Pause/continue source step.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>Q</td>
<td>Close Model Debugger.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>R</td>
<td>Restart simulation.</td>
</tr>
<tr>
<td>Ctrl + Shift</td>
<td>R</td>
<td>Reset target.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>S</td>
<td>Save the current session.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>T</td>
<td>Select target.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>U</td>
<td>User preferences dialog.</td>
</tr>
<tr>
<td>Ctrl</td>
<td>W</td>
<td>Close model.</td>
</tr>
</tbody>
</table>