

# Model Debugger for Fast Models

Version 8.1

## User Guide



# Model Debugger for Fast Models

## User Guide

2007-2013

### Release Information

### Document History

Issue	Date	Confidentiality	Change
A	31 August 2007	Confidential	Release for System Generator v3.0
B	31 December 2007	Confidential	Update for System Generator v3.1
C	29 February 2008	Confidential	Update for System Generator v3.2
D	31 March 2008	Confidential	Update for SoC Designer v7.1
E	30 June 2008	Confidential	Release for System Generator v4.0
F	31 August 2008	Confidential	Update for System Generator v4.0 SP1
G	31 December 2008	Confidential	Update for Fast Models v4.1
H	31 March 2009	Non-Confidential	Update for Fast Models v4.2
I	30 April 2009	Non-Confidential	Release for Fast Models v5.0
J	30 September 2009	Non-Confidential	Update for Fast Models v5.1
K	28 February 2010	Non-Confidential	Update for Fast Models v5.2
L	31 October 2010	Non-Confidential	Release for Fast Models v6.0
M	30 November 2011	Non-Confidential	Release for Fast Models v7.0
N	31 December 2012	Non-Confidential	Release for Fast Models v8.0
O	31 May 2013	Non-Confidential	Update for Fast Models v8.1

### Proprietary Notice

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

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

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

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

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

### Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Unrestricted Access is an ARM internal classification.

### Product Status

The information in this document is Final, that is for a developed product.

### Web Address

[www.arm.com](http://www.arm.com)

# Contents

## Model Debugger for Fast Models User Guide

	<b>Preface</b>	
	About this book .....	8
	Feedback .....	9
<b>Chapter 1</b>	<b>Introduction to Model Debugger</b>	
	1.1 About Model Debugger .....	1-11
<b>Chapter 2</b>	<b>Using Model Debugger</b>	
	2.1 Launching Model Debugger .....	2-15
	2.2 Model Debugger application windows .....	2-26
	2.3 Debug views for source code and disassembly .....	2-42
	2.4 Debug views for registers and memory .....	2-51
	2.5 Debug views for pipelines .....	2-58
	2.6 Watch window and Expression Evaluator .....	2-63
	2.7 Setting breakpoints in the debug views .....	2-66
	2.8 Model Debugger sessions .....	2-71
	2.9 Preferences dialog box .....	2-72
<b>Chapter 3</b>	<b>Installation and Configuration</b>	
	3.1 Linux installation procedure .....	3-75
	3.2 Windows installation procedure .....	3-77
<b>Chapter 4</b>	<b>Shortcuts</b>	
	4.1 Keyboard shortcuts .....	4-79

# List of Figures

## Model Debugger for Fast Models User Guide

Figure 1-1	A Model Debugger session: main window with debug windows .....	1-11
Figure 2-1	Debug Simulation dialog box .....	2-19
Figure 2	Configure Model Parameters dialog box .....	20
Figure 3	Configure Model Parameters dialog box, List View tab .....	21
Figure 4	Select Targets dialog box .....	22
Figure 5	Load Application dialog box .....	23
Figure 2-6	Connect remote dialog box .....	2-24
Figure 2-7	Select Target dialog box .....	2-24
Figure 2-8	Debug Isim System dialog box .....	2-25
Figure 2-9	Default layout for Model Debugger .....	2-27
Figure 2-10	Main toolbar .....	2-28
Figure 2-11	Configure cores for MP stepping dialog box .....	2-33
Figure 2-12	Drag-and-drop of debug views, while moving the Memory window .....	2-36
Figure 2-13	Duplicating a register view .....	2-37
Figure 2-14	Layout Control window .....	2-38
Figure 2-15	Layout Control context menu .....	2-38
Figure 2-16	Load Layout dialog box .....	2-39
Figure 2-17	Icons for selecting a new debug view .....	2-40
Figure 2-18	Closing windows or individual debug views .....	2-40
Figure 2-19	Arrow button for scrolling code .....	2-42
Figure 2-20	Source view .....	2-43
Figure 2-21	Debug Source Files dialog box .....	2-44
Figure 2-22	Find Source File dialog box .....	2-45

Figure 2-23	Source File Properties dialog box .....	2-45
Figure 2-24	Source Path Replacement dialog box .....	2-46
Figure 2-25	Find dialog box .....	2-47
Figure 2-26	Disassembly view .....	2-48
Figure 2-27	Matching source and disassembly .....	2-49
Figure 2-28	Call Stack view .....	2-50
Figure 2-29	Select register group .....	2-51
Figure 2-30	Register view showing current and previous contents .....	2-51
Figure 2-31	Register view contents at cursor .....	2-52
Figure 2-32	Memory view .....	2-53
Figure 2-33	Load File to Memory dialog box .....	2-55
Figure 2-34	Local Variable view .....	2-56
Figure 2-35	Global Variable view .....	2-56
Figure 2-36	Pipeline Overview window .....	2-58
Figure 2-37	Pipeline Table window .....	2-59
Figure 2-38	Pipeline Table icons .....	2-59
Figure 2-39	Pipeline Table context menu .....	2-60
Figure 2-40	Pipeline Stage Properties dialog box .....	2-61
Figure 2-41	Pipeline view context menu .....	2-61
Figure 2-42	Submenu for display format .....	2-62
Figure 2-43	Watch window .....	2-63
Figure 2-44	Source view breakpoint .....	2-66
Figure 2-45	Disassembly view breakpoint .....	2-66
Figure 2-46	Register view breakpoint .....	2-67
Figure 2-47	Pipeline table breakpoint .....	2-67
Figure 2-48	Breakpoint Manager dialog box .....	2-69
Figure 2-49	Breakpoint Properties dialog box .....	2-70
Figure 2-50	Preferences dialog box .....	2-72

# List of Tables

## Model Debugger for Fast Models User Guide

<i>Table 2-1</i>	<i>Command line options .....</i>	<i>2-15</i>
<i>Table 2-2</i>	<i>Operator precedence .....</i>	<i>2-64</i>
<i>Table 3-1</i>	<i>Environment variables .....</i>	<i>3-76</i>
<i>Table 4-1</i>	<i>Keyboard shortcuts .....</i>	<i>4-79</i>

# Preface

This preface introduces the *Model Debugger for Fast Models User Guide*.

It contains the following sections:

- *About this book on page 8.*
- *Feedback on page 9.*

## About this book

ARM Model Debugger User Guide. This is the reference manual that documents how to use Model Debugger GUI for CADI-compliant processor models generated using the System Generator for ARM Fast Models tool.

## Using this book

This book is organized into the following chapters:

**Chapter 1 Introduction to Model Debugger**

**Chapter 2 Using Model Debugger**

**Chapter 3 Installation and Configuration**

**Chapter 4 Shortcuts**

## Typographic conventions

<i>italic</i>	Introduces special terminology, denotes cross-references, and citations.
<b>bold</b>	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
<code>monospace</code>	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<u><code>monospace</code></u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<i><code>monospace</code></i> <i>italic</i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
<code>monospace</code> <b>bold</b>	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: <code>MRC p15, 0 &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;</code>
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>ARM glossary</i> . 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 <mailto:errata@arm.com>. Give:

- The title.
- The number ARM DUI0314O.
- The page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

# Chapter 1

## Introduction to Model Debugger

This chapter describes the main features of Model Debugger.

It contains the following sections:

- *About Model Debugger on page 1-11.*

## 1.1 About Model Debugger

Model Debugger for Fast Models is a fully retargetable debugger for scalable multiprocessor software development. It is 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.

Model Debugger can connect to any Component Architecture Debug Interface (CADI) compliant models.

Full multiprocessor debugging support is included and multiple instances of Model Debugger stay fully synchronized while debugging different processors running within a single system.

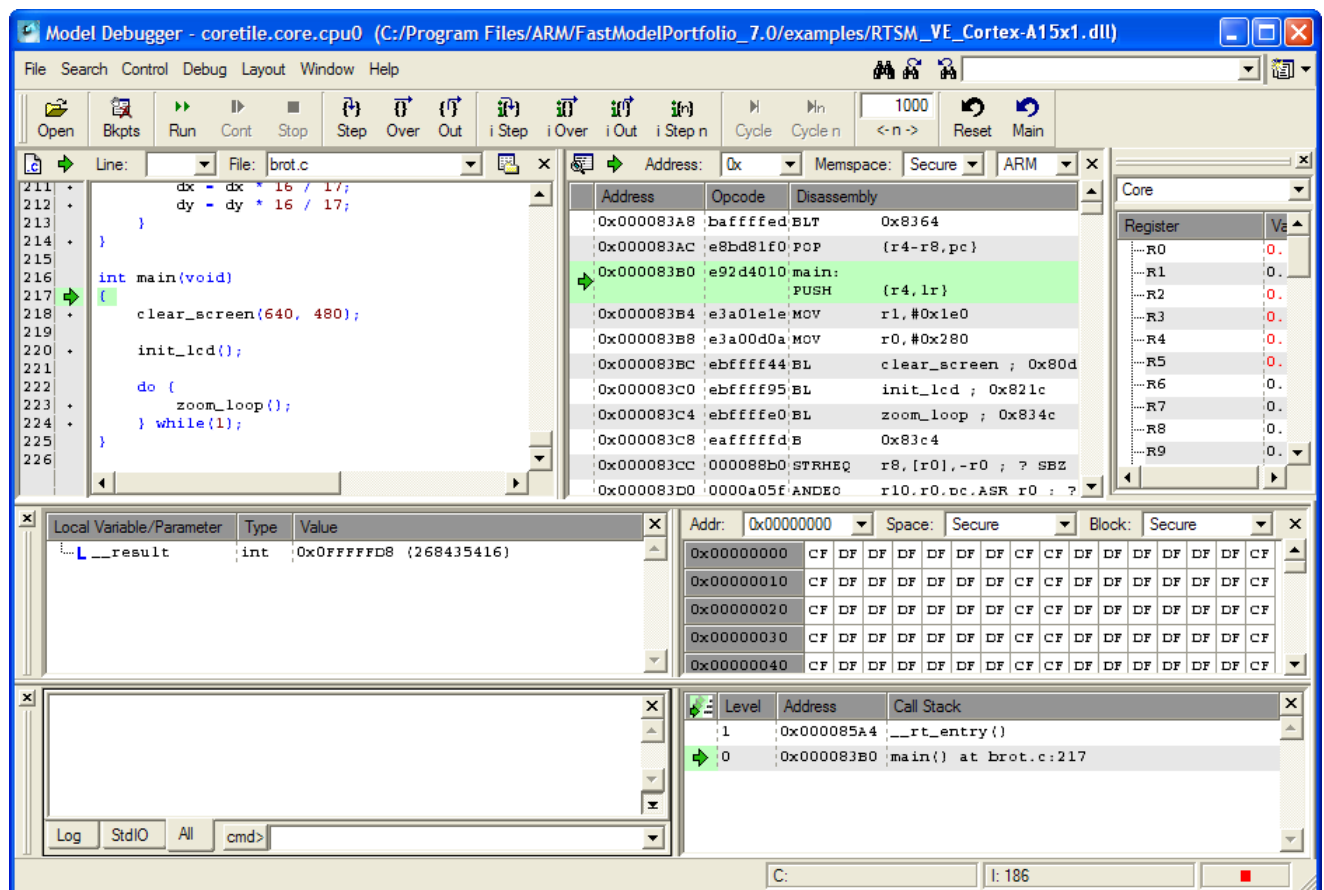


Figure 1-1 A Model Debugger session: main window with debug windows

It contains the following sections:

- [Key features on page 1-12.](#)
- [Retargetable debugger on page 1-12.](#)
- [Multiprocessor debugging on page 1-12.](#)

### 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.
- Advanced search capabilities.
- Intuitive GUI with fully customizable window layout.
- Project management to store debugging sessions including window layout, open files and breakpoints.

---

**Note**

Not all of these features are supported by Fast Models targets.

---

### 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 to retrieve the static target specific information, for example a register file, to determine the target state and to control execution.

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

#### Related Information

[Component Architecture Debug Interface v2.0 Developer Guide.](#)

### 1.1.3 Multiprocessor debugging

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

If attached to a processor 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.

#### Related References

[Model Debugger sessions on page 1-71.](#)

## Related Information

*[Component Architecture Debug Interface v2.0 Developer Guide.](#)*

# Chapter 2

## Using Model Debugger

This chapter describes how to use Model Debugger.

It contains the following sections:

- *Launching Model Debugger on page 2-15.*
- *Model Debugger application windows on page 2-26.*
- *Debug views for source code and disassembly on page 2-42.*
- *Debug views for registers and memory on page 2-51.*
- *Debug views for pipelines on page 2-58.*
- *Watch window and Expression Evaluator on page 2-63.*
- *Setting breakpoints in the debug views on page 2-66.*
- *Model Debugger sessions on page 2-71.*
- *Preferences dialog box on page 2-72.*

## 2.1 Launching Model Debugger

Launch Model Debugger from the command line or from within System Canvas. If required, start Model Debugger independently of these GUI tools.

Use the Preferences dialog box to modify the preferences of Model Debugger after it has started.

———— **Note** ————

Model Debugger supports debugging of multiple components.

It contains the following sections:

- [Using the command line on page 2-15.](#)
- [Launching from System Canvas on page 2-18.](#)
- [Launching Model Debugger separately on page 2-23.](#)
- [Start Simulation and Connect automatically on page 2-25.](#)

### 2.1.1 Using the command line

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

The following table lists the arguments and options that can be passed on the command line.

**Table 2-1 Command line options**

Short	Long option	Description
	<code>--cyclelimit <i>cycles</i></code>	Set a limit on the number of system cycles for a simulation in non-GUI mode. The cyclelimit option is only enabled if the <code>--nogui</code> option is used.
	<code>--debug-isim <i>isim_system</i></code>	Start <i>isim_system</i> and connect Model Debugger remote to the simulation.
	<code>--debug-sysc <i>systemc</i></code>	Start <i>systemc</i> Simulation and connect Model Debugger remote to the simulation.
	<code>--timelimit <i>time</i></code>	Set a time limit for a simulation in non-GUI mode. The timelimit option is only enabled if the <code>--nogui</code> option is used.
-a	<code>--application <i>filename</i></code>	Load the application file <i>filename</i> . To target processors in multiprocessor systems, prefix the name with the path to the instance. For example, <code>foo.bar.core=dhrystone.axf</code> .
-C	<code>--parameter <i>parameter</i></code>	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.
-c	<code>--connect <i>simulation_id</i></code>	Connect to a remote CADI simulation. The simulation to connect to is specified by the <i>simulation_id</i> . Use the <code>--list-connections</code> option to display the list of available connections.

**Table 2-1 Command line options (continued)**

Short	Long option	Description
-E	--enable-verbose <i>msgClass</i>	Use verbose messages if displaying message text for message classes <i>msgClass</i> . if used without an argument, lists all classes.
-e	--env-connect	Connect to remote CADI simulation using the following environment variables: <ul style="list-style-type: none"> <li>CADI_CLIENTPORT_TCP – port number</li> <li>CADI_INSTANCEID – component instance name</li> <li>CADI_APPLICATIONFILENAME – application file name</li> </ul>
-F	--stdout-to-file <i>FILE</i>	Print all application output to <i>FILE</i> instead of the StdIO pane in the Output Window.
-f	--config-file <i>filename</i>	Use model parameters from the configuration file <i>filename</i> .
-h	--help	Print the available options and exit.
-i	--instance	Specify the instance.
-L	--cadi-log	Log all CADI calls into an XML logfile <i>CADILog-<i>nnnn</i>.xml</i> , where <i>nnnn</i> is a set of four digits. The logfile is created in the same directory as the model.
-l	--list-connections	List possible connections to remote CADI simulations on the local machine and exit afterwards. Each simulation is given a unique simulation ID.
	--list-instances	List target instances.
	--list-params	List target instances and their parameters.
-m	--model <i>filename</i>	Load the model library in the file named <i>filename</i> .
-N	--nogui	Run the simulation without displaying the GUI. Use the --script option to load and run a script on startup.
-n	--no-params-dialog	Do not display the parameter configuration dialog at startup.
-O	--stdout-to-stdout	Print all application output to <i>stdout</i> instead of the StdIO pane in the Output Window.
-p	--project <i>filename</i>	Load the project file <i>filename</i> .
-q	--quiet	Suppress all Model Debugger and Model Shell output.
-s	--script <i>filename</i>	Execute the commands from the script named <i>filename</i> .
	--trace-plugin	Specify trace plugins. All plugins specified with this command line option or in environment variable <i>FM_TRACE_PLUGINS</i> are loaded.
-V	--verbose	Equivalent to --enable-verbose “MaxView”.
-v	--version	Print the tool version and exit.



Table 2-1 Command line options (continued)

Short	Long option	Description
-x	--force-reg-hex	Force registers with initial integer display to be hexadecimal format instead.
-Y	--layout <i>filename</i>	Load the layout file <i>filename</i> .
-y	--no-target-dialog	Suppress the Select Target dialog box that normally appears when a model is loaded. Model Debugger automatically connects to targets that have the <code>executes_software</code> flag set. From the GUI, you can use the <b>Other Settings</b> check box in the Preferences dialog box to suppress the Select Target dialog box.

### 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.

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
```

### Related References

[String syntax on page 17.](#)

### 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.

### 2.1.2 Launching from System Canvas

To launch Model Debugger from System Canvas, either:

- Click the **Debug** button on the System Canvas toolbar.
- In the main System Canvas menu, select **Project > Launch Model Debugger**.

The Debug Simulation dialog box appears:

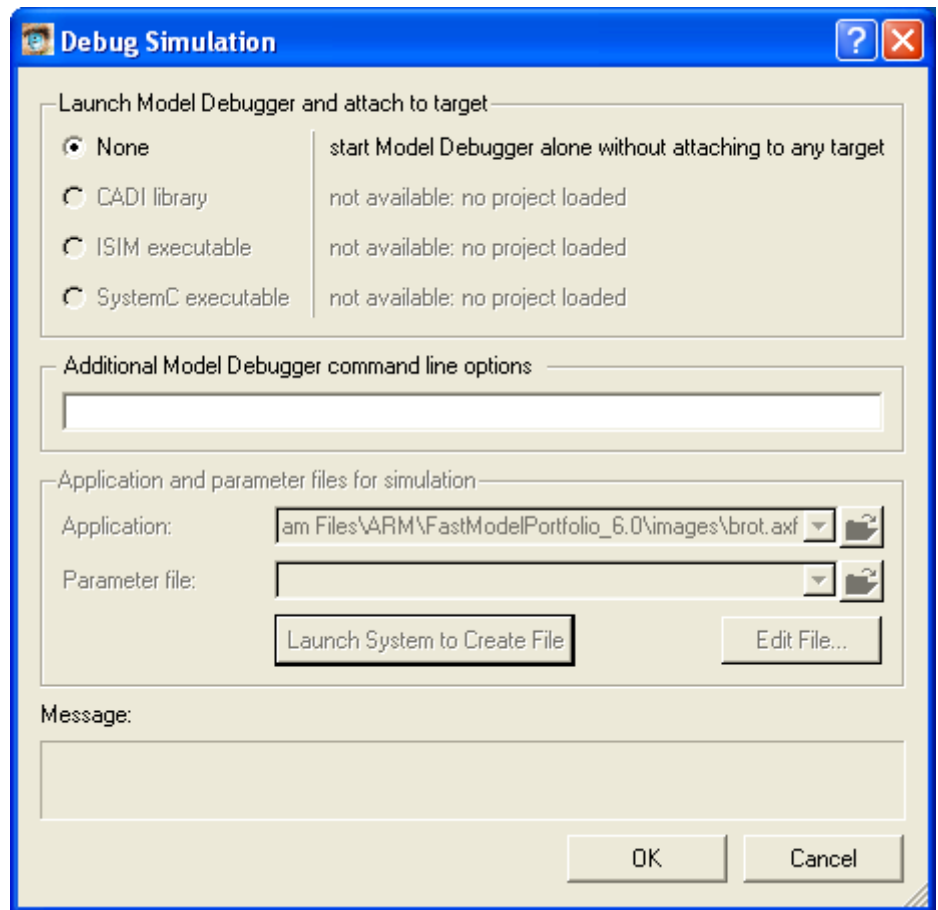


Figure 2-1 Debug Simulation dialog box

———— **Note** ————

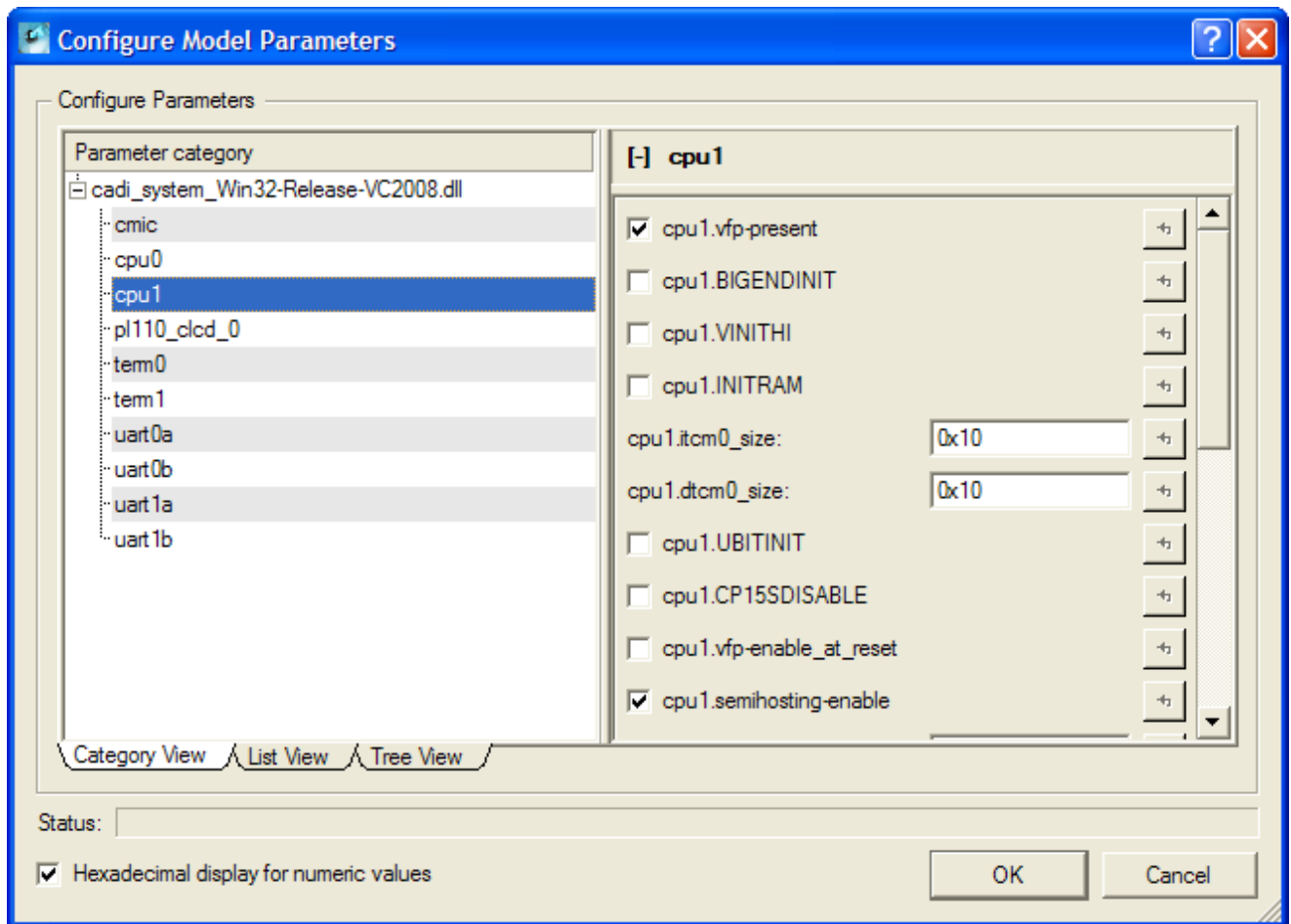
If you have loaded a model, the **CADI library** option and the **Application** field are available for use.

Click **OK**, and Model Debugger starts.

### Using the Configure Model Parameters dialog box

If you had not yet loaded a model at the time that Model Debugger starts:

1. Select **File > Load Model**.
2. In the Load Model dialog box that appears, locate and select the required model, then click **Open**.
3. A dialog box appears:



**Figure 2 Configure Model Parameters dialog box**

If you had already loaded a model before Model Debugger starts, Model Debugger checks the available components and opens a similar dialog box.

———— **Note** ————

The exact contents and titles of the panes might differ because they depend on the model.

The Configure Model Parameters dialog box has the following sections:

- |                           |  |
|---------------------------|--|
| <b>Parameter category</b> | 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.      |
| <b>Parameter setting</b>  | The parameter values are displayed in the right-hand pane.<br>Use the box in the lower left corner of the window to toggle between hexadecimal or decimal views. |

———— **Note** ————

The name of this pane varies, depending on the model, but its purpose and usage is the same in all cases.

To view all of the configuration parameters as a single list, click the **List View** tab. The **Tree View** is similar, but shows the same parameters in a grouped hierarchy.

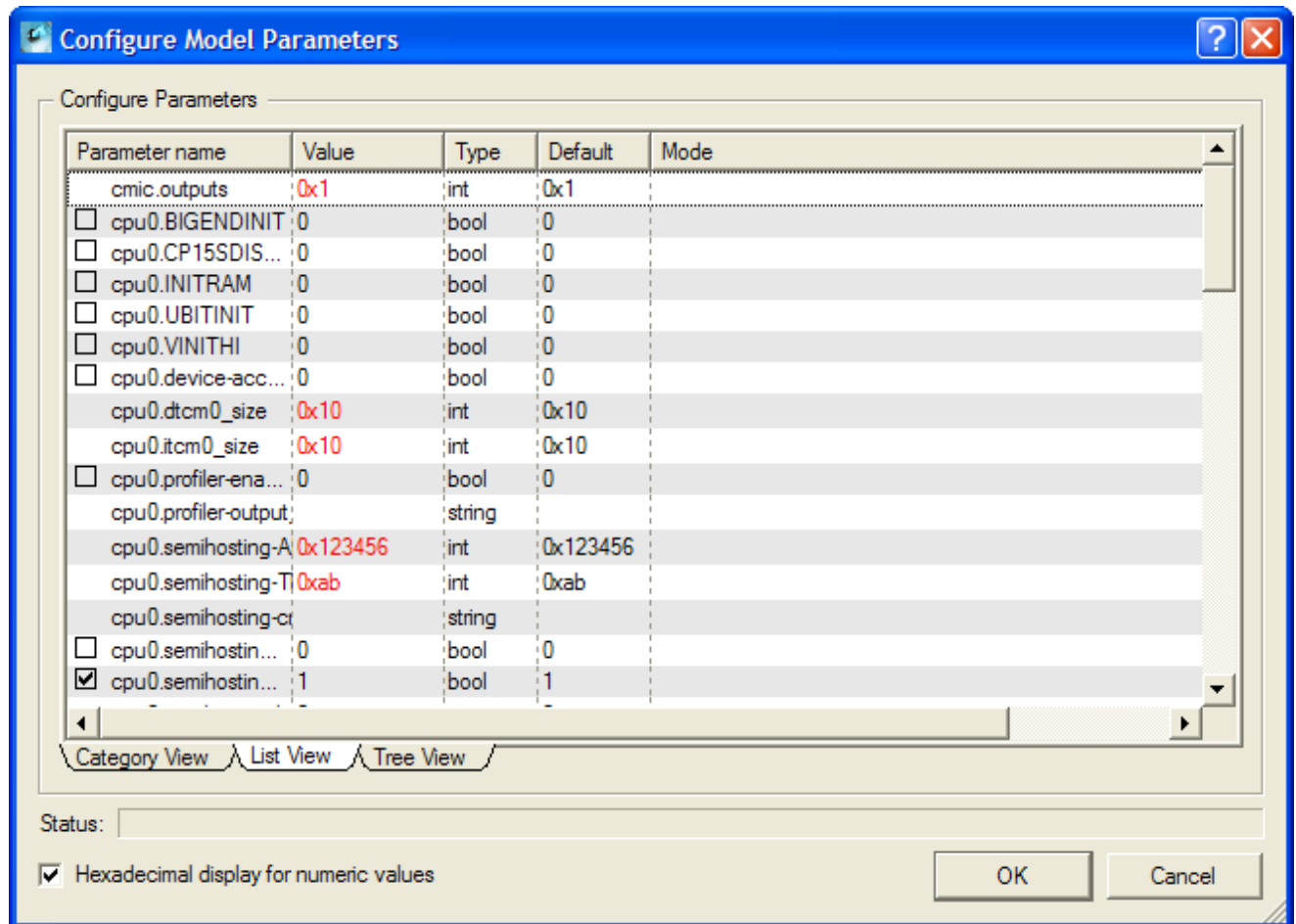


Figure 3 Configure Model Parameters dialog box, List View tab

Set the parameters for the model and click **OK** to close the Configure Model Parameters dialog box.

### Using the Select Targets dialog box

After closing the Configure Model Parameters dialog box, the Select Targets dialog box appears. Select those components that can be debugged in the model.

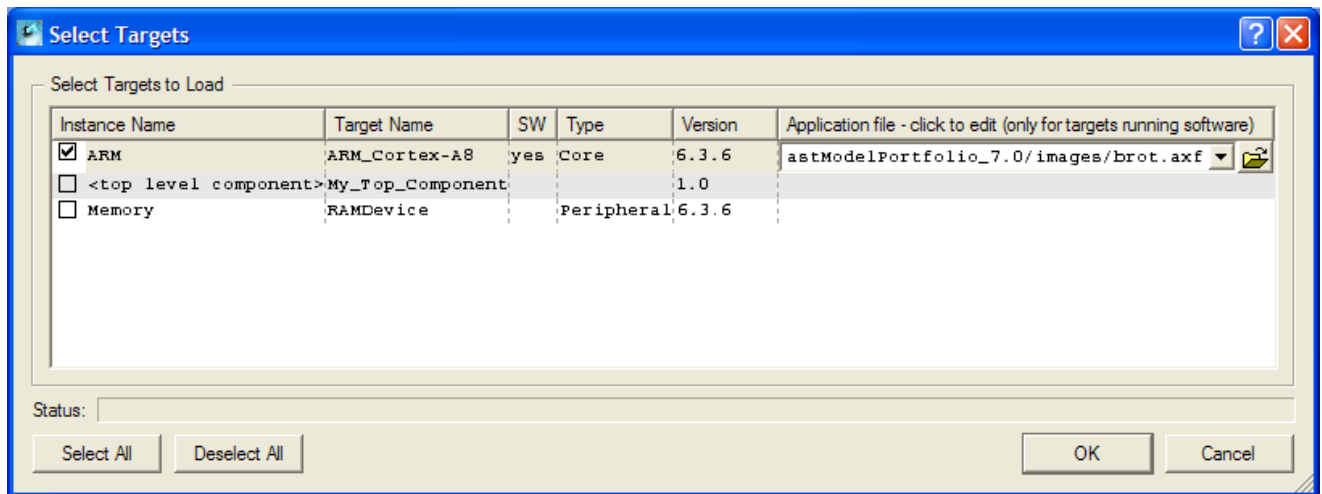


Figure 4 Select Targets dialog box

1. Click the box next to the components that are to load.
2. The **Application file** column displays applications to load for the processors. 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 box.
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 box to specify the location for the source code.

### Using the Load Application dialog box

If the application and source code are not loaded automatically, select **File > Load Application** to locate and load the code manually.

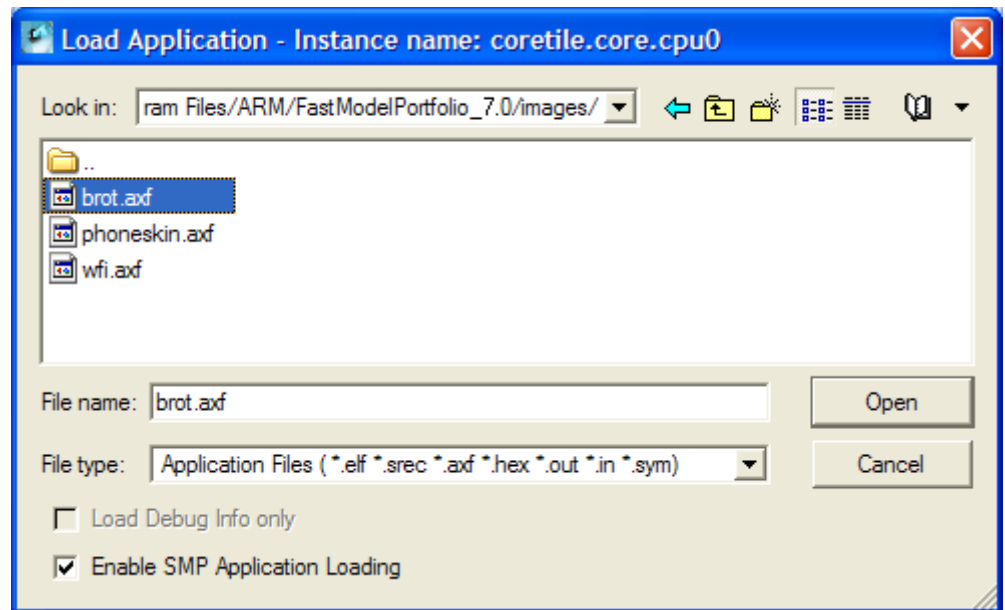


Figure 5 Load Application dialog box

Reset the component after loading the source.

———— **Note** ————

The Load Application dialog box only displays if you have loaded a model from Model Debugger.

### 2.1.3 Launching Model Debugger separately

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

#### Procedure

1. Start your model, for example using Model Shell with a CADI server enabled.
2. Launch Model Debugger.
3. Select **File > Connect to Model** to display the Model Debugger - Connect remote dialog box:

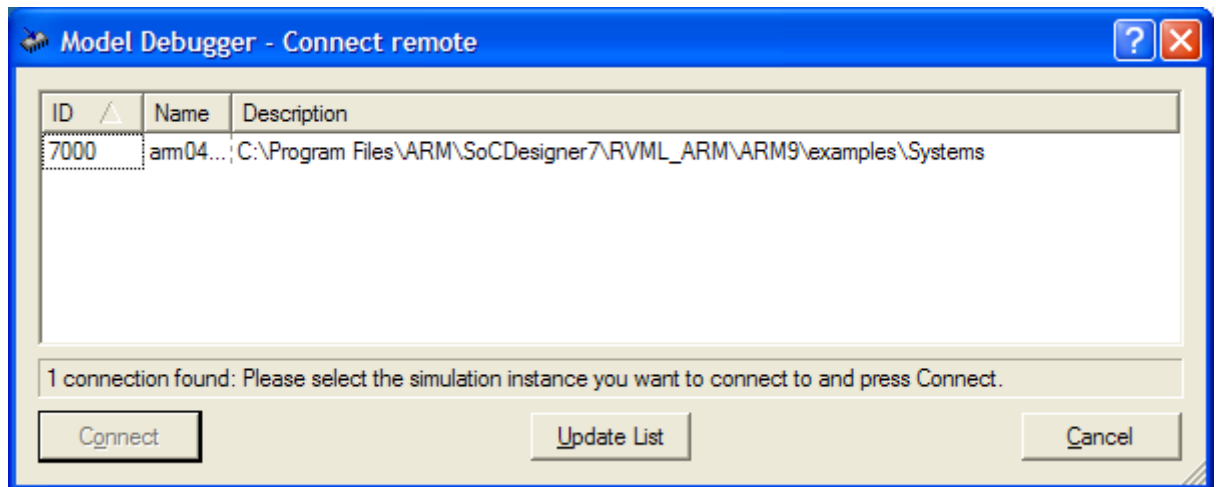


Figure 2-6 Connect remote dialog box

4. Select the required simulation instance and click **Connect**.
5. Select a target, for example the processor, and click **Connect** to connect to the specific component instance in the simulation.

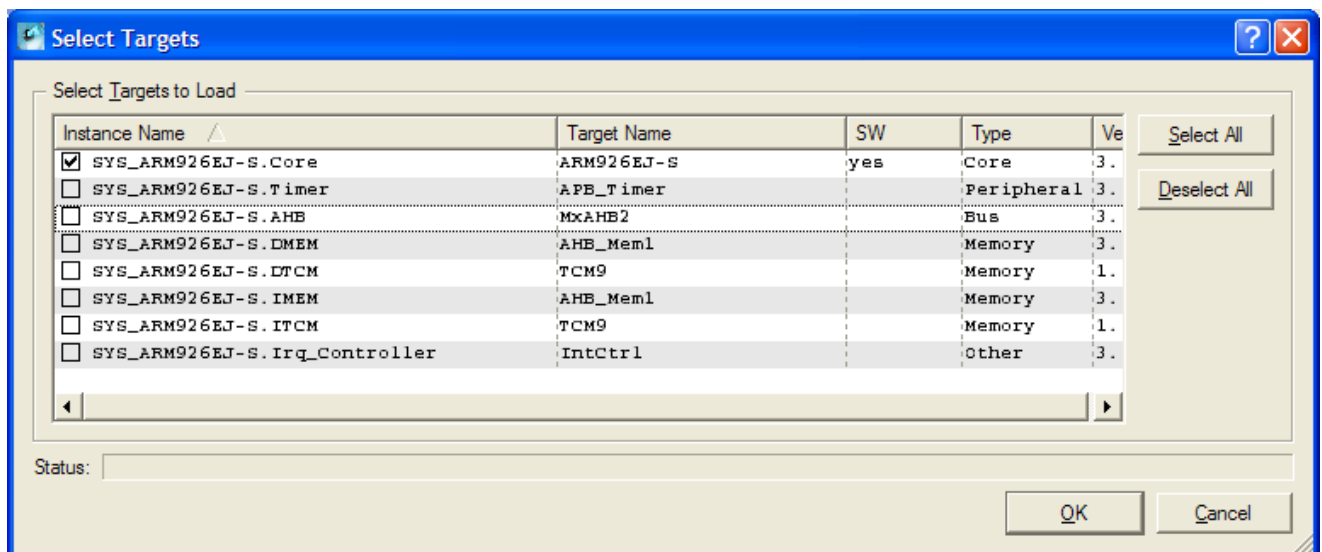


Figure 2-7 Select Target dialog box

If you select more than one instance, one Model Debugger window is opened for each component. Click **OK** to close the dialog box.

6. If no application is loaded, select **File > Load Application Code**. Select the application image from the Load Application dialog box 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.

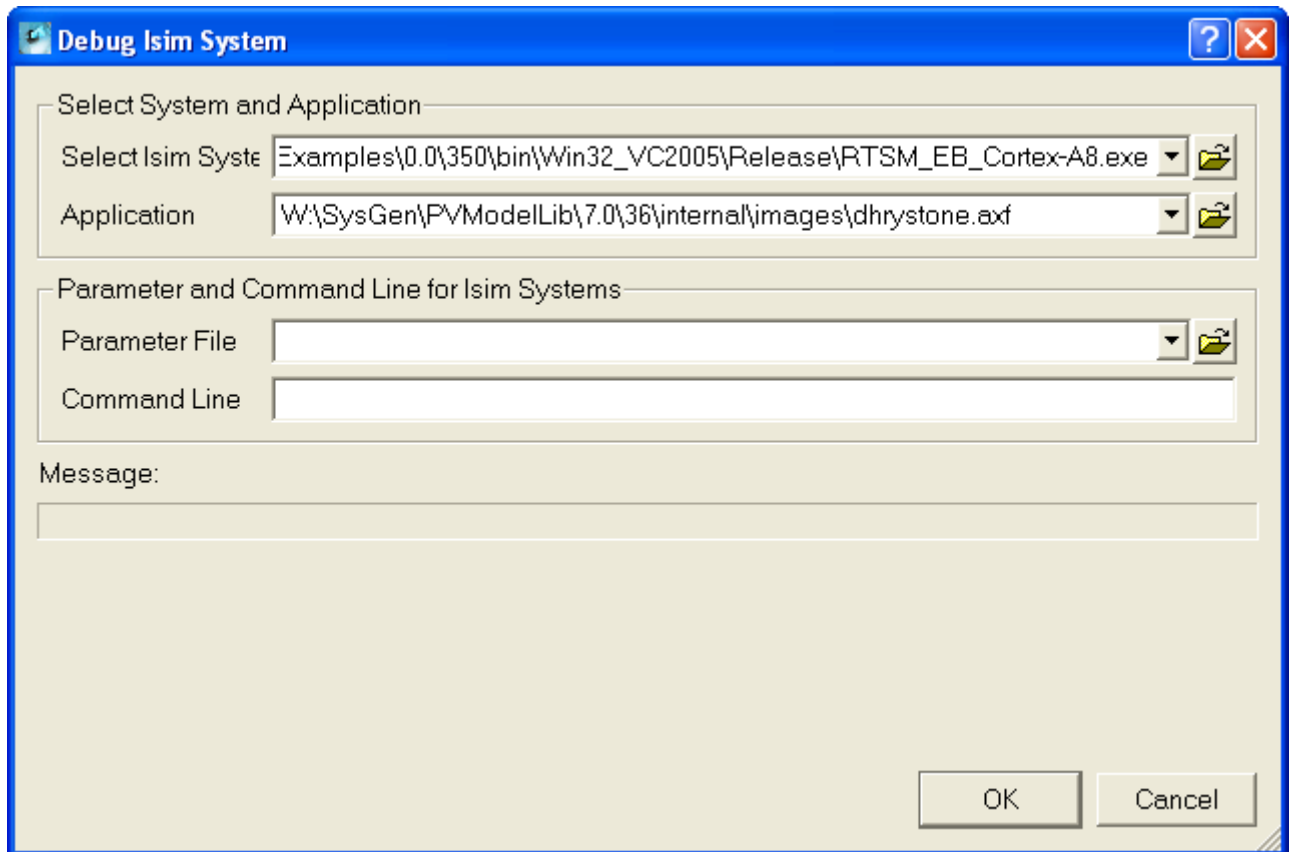


### 2.1.4 Start Simulation and Connect automatically

Model Debugger can start a SystemC or isim model simulation and connect to it automatically after it has started. To do this:

#### Procedure

1. In Model Debugger, either:
  - select **File > Debug Isim System ...** to display the Debug Isim System dialog box
  - select **File > Debug SystemC Simulation ...** to display the Debug SystemC Simulation dialog box.



**Figure 2-8 Debug Isim System dialog box**

2. Select a simulation and optionally an application (and parameter file for isim only). An error message appears if one of the files cannot be found. Click **OK** to start the simulation and connect.
3. The Select Targets dialog box appears.

#### Related Tasks

*Launching Model Debugger separately on page 2-23.*

#### Related References

*Preferences dialog box on page 2-72.*

## 2.2 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 box.

---

**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.

---

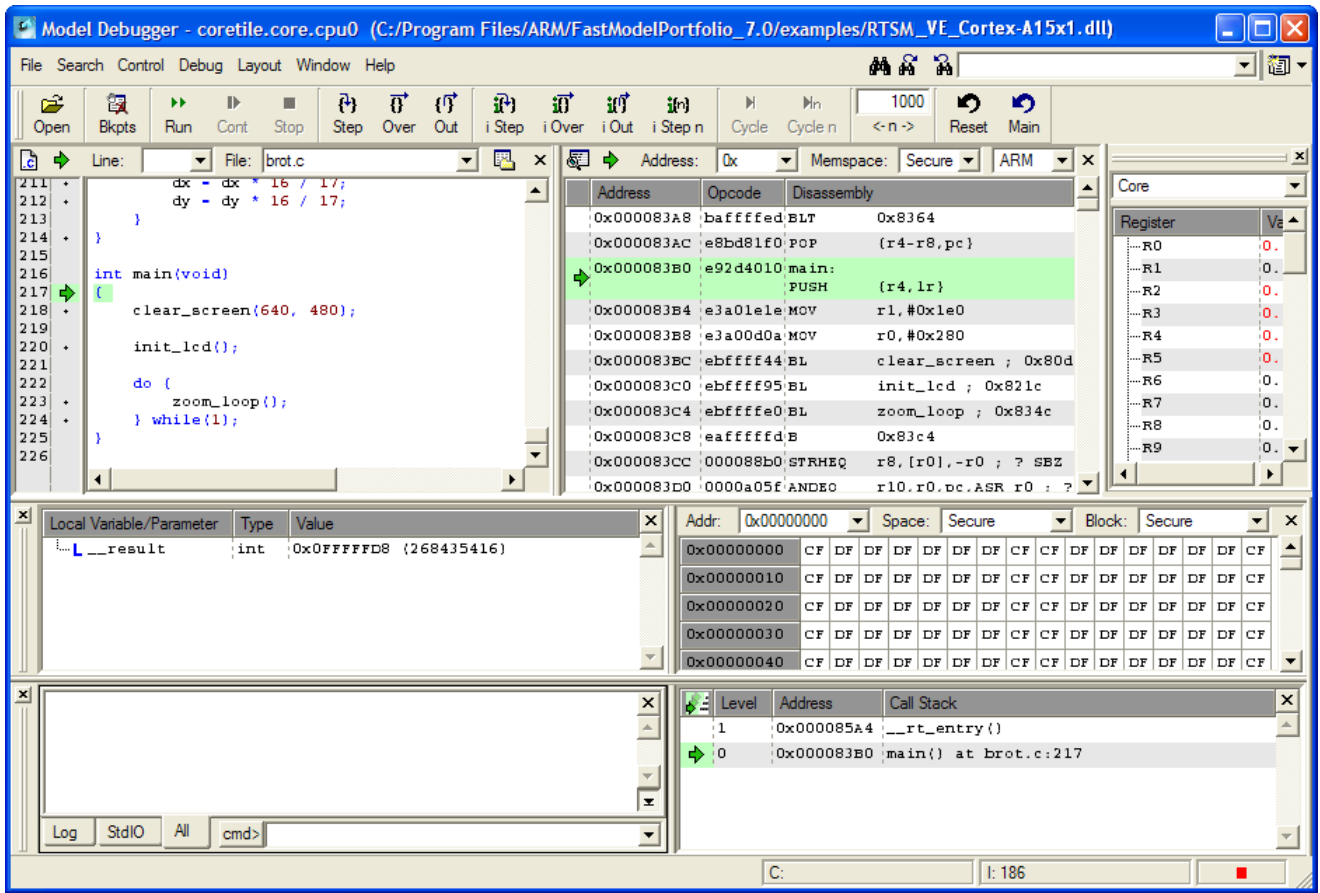


Figure 2-9 Default layout for Model Debugger

It contains the following sections:

- *Main toolbar on page 2-27.*
- *Menu bar on page 2-30.*
- *Dock windows on page 2-35.*
- *Moving or copying views on page 2-35.*
- *Saving the window layout on page 2-37.*
- *Opening new debug views on page 2-39.*
- *Closing windows and views on page 2-40.*
- *Output window on page 2-40.*

### 2.2.1 Main toolbar

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

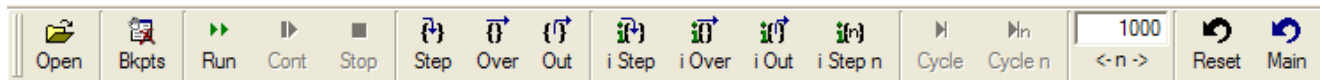


Figure 2-10 Main toolbar

### 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 box 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 box 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 box is displayed to select the source file for the application.

Select the source file and click **OK**.

### ———— Note ————

If you are using a Symmetric MultiProcessing (SMP) model with more than one processor, such as one based on the Cortex™-A9, then Model Debugger only loads one image that is run on all processors. 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 box.

### 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.

<b>i Over</b>	Click to advance the simulation by one source-level instruction without following any call instructions.  ———— <b>Note</b> ———— This command is not supported by all model targets.  —————
<b>i Out</b>	Click to advance the simulation until a return instruction is executed.  ———— <b>Note</b> ———— This command is not supported by all model targets.  —————
<b>i Step n</b>	Click to advance the simulation by executing the number of source-level instructions specified in the <-n-> control.
<b>Cycle</b>	Click to advance the simulation by a single cycle.
<b>Cycle n</b>	Click to advance the simulation by the number of cycles specified in the edit box. The default is 1000 cycles.
<b>&lt;-n -&gt;</b>	Enter the number of cycles to step if the <b>Cycle n</b> or <b>Back n</b> buttons are clicked. The default is 1000 cycles.  If the <b>i Step n</b> button is clicked, this control indicates the number of instructions to step.
<b>Back n</b>	Click to step the simulation backwards by the number of cycles specified in the edit box. The default is 1000 cycles.  ———— <b>Note</b> ———— This command is not supported by all model targets.  —————
<b>Back</b>	Click to step the simulation backwards by one cycle.  ———— <b>Note</b> ———— This command is not supported by all model targets.  —————
<b>Reset</b>	Click to cause a reset of the target model. The application is reloaded.
<b>Main</b>	Click to cause a reset of the target model. The application is reloaded. The model runs until the <code>main()</code> function of the application source code is reached.  ———— <b>Note</b> ———— This command is only available if a function <code>main()</code> can be found in the debug information of the application file.  —————

## Related References

*Preferences dialog box on page 2-72.*

## 2.2.2 Menu bar

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

### File menu

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

<b>Open Source ...</b>	Opens the source code for the application.
<b>Source File Manager ...</b>	Displays the Source File Manager dialog box.
<b>Load Application Code ...</b>	Loads application code to the model.
<b>Load Application Code (Debug info only) ...</b>	Loads debug information only.
<b>Load Model ...</b>	Loads a model.
<b>Connect to Model ...</b>	Displays the Connect to Target dialog box to connect to a model file.
<b>Debug Isim System ...</b>	Displays the Debug Isim System dialog box to start and debug an isim system.
<b>Debug SystemC Simulation ...</b>	Displays the Debug SystemC Simulation dialog box to start and debug a SystemC simulation.
<b>Close Model</b>	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.
<b>Open Session ...</b>	Opens a previously saved session.
<b>Save Session</b>	Saves the current debug session.
<b>Save Session As</b>	Saves the current debug session to a new location and name.
<b>Preferences</b>	Displays the Preferences dialog box to enable you to modify the user preferences.
<b>Recently Opened Models</b>	<p>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 box.</p> <p>To remove a file from the list, move the mouse cursor over the file name and press the <b>Delete</b> key or right click and select <b>Remove from list</b> from the context menu.</p>

<b>Recently Opened Applications</b>	<p>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 box.</p> <p>To remove a application from the list, move the mouse cursor over the application name and press the <b>Delete</b> key or right click and select <b>Remove from list</b> from the context menu.</p>
<b>Recently Opened Sessions</b>	<p>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 box.</p> <p>To remove a session from the list, move the mouse cursor over the session name and press the <b>Delete</b> key or right click and select <b>Remove from list</b> from the context menu.</p>
<b>Exit</b>	<p>Ends Model Debugger. If you have modified files or sessions, a dialog box prompts you to save your changes.</p>

## Search menu

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

<b>Find ...</b>	Opens a dialog box that enables searching for a string in a currently active window.
<b>Find Next</b>	Repeats the last defined search to find the next occurrence.
<b>Find Previous</b>	Repeats the last defined search, but the search direction is backwards in the document.

## Control menu

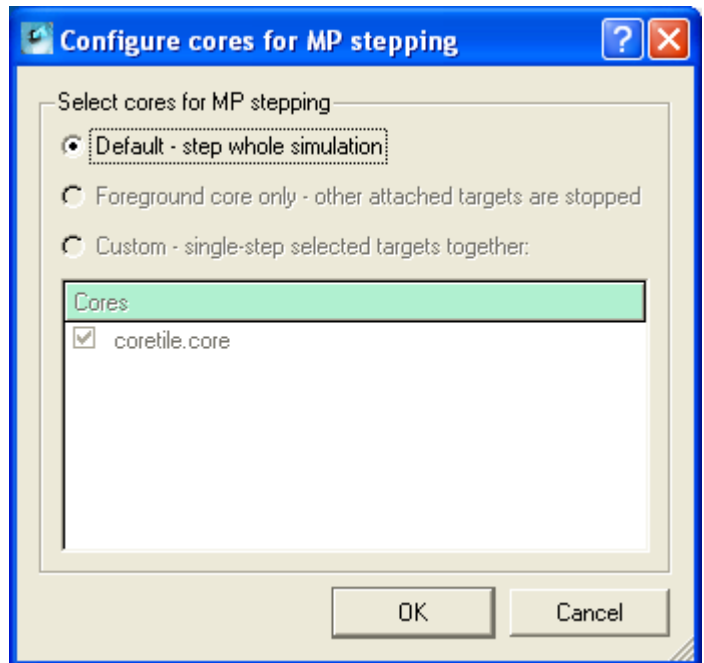
The **Control** menu has the following options:

<b>Hard Reset</b>	This option resets the simulation without reloading the application.
<b>Reset</b>	Click to cause a reset of the target model. The application is reloaded automatically.
<b>Goto Main</b>	Cause a reset of the target model. The application is reloaded. The model runs until the <code>main()</code> function of the application source code is reached.
<p>———— <b>Note</b> —————</p> <p>This command is only available if a function <code>main()</code> can be found in the debug information of the application file.</p> <p>—————</p>	
<b>Run</b>	Run the simulation until a breakpoint is hit or some exception occurs. An example would be simulation halt.
<b>Pause/Continue Source Step</b>	Pause or continue the current high-level simulation step command. An example would be a source-level step.

<b>Source Step Over</b>	Cause a source-level step to execute until the simulation reaches a different source line.
<b>Source Step Out</b>	Cause source-level steps to execute control command until the current function is exited.
<b>Instruction Step</b>	Advance the simulation by executing one source-level instruction.
<b>Instruction Step Over</b>	Advance the simulation by one source-level instruction without following any call instructions.
<p>———— <b>Note</b> ————</p> <p>This command is not supported by all model targets.</p> <p>—————</p>	
<b>Instruction Step Out</b>	Advance the simulation until a return instruction is executed.
<p>———— <b>Note</b> ————</p> <p>This command is not supported by all model targets.</p> <p>—————</p>	
<b>Instruction Step n</b>	Advance the simulation by the number of instructions specified in the <- n -> edit box. The default is 1000 cycles.
<b>Cycle Step</b>	Advance the simulation by a single cycle.
<b>Cycle Step n</b>	Advance the simulation by the number of cycles specified in the edit box. The default is 1000 cycles.
<b>Enable/Disable Step Back</b>	Enable or disable stepping back by cycles.
<p>———— <b>Note</b> ————</p> <p>This command is not supported by all model targets.</p> <p>—————</p>	
<b>Back</b>	Step the simulation backwards by one cycle.
<p>———— <b>Note</b> ————</p> <p>This command is not supported by all model targets.</p> <p>—————</p>	
<b>Back n</b>	Step the simulation backwards by the number of cycles specified in the edit box. The default is 1000 cycles.
<p>———— <b>Note</b> ————</p> <p>This command is not supported by all model targets.</p> <p>—————</p>	



**Configure cores for MP stepping ...** Use the **Configure cores for MP stepping** dialog box to enable independent execution of processors, that is, targets.



**Figure 2-11 Configure cores for MP stepping dialog box**

In multiprocessor debugging, each Model Debugger window is connected to a particular target, and the controls in that window apply only to that target. It is the simulation that determines how other connected targets behave when you click **Stop**, **Step** or **Run** within a window. Typical behavior is to stop and run the whole simulation.

You use the **Configure cores for MP stepping** dialog box to enable Model Debugger to override the default behavior. Model Debugger can control each target to which it is connected, and can force that target to stop executing code while the simulation is running or stepping. In that instance, Model Debugger does not stop any target to which it is not connected, so you must connect to a target if you want it to stop during independent stepping, even if you do not specifically want to view or control that target.

———— **Note** ————

The **Configure cores for MP stepping** dialog box is only enabled if you have loaded a model.

The available MP stepping modes are as follows:

- Use **Default - step whole simulation** to place all execution control with the simulator. In this mode, Model Debugger does not explicitly stop any targets.
- Use **Foreground core only - other attached targets are stopped** to cause Model Debugger to enable the foreground target to run, and to stop all other targets to which it is connected.

---

**Note**

The foreground target is the target associated with the window that you have selected to run.

- 
- Use **Custom - single-step selected targets together** to cause Model Debugger to enable a fixed set of targets to run, and to stop all other targets to which Model Debugger is connected. In this mode, the user interface disables any step or run controls for deselected targets.

## Debug menu

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

<b>Display Messages</b>	Display debug messages.
<b>Clear Log</b>	Clear the log of debug messages.
<b>Clear Model Output</b>	Clears all output messages from the model.
<b>Clear Output Summary</b>	Clear the summary output messages.
<b>Breakpoint Manager ...</b>	Display the Breakpoint Manager dialog box.
<b>Profiling Manager ...</b>	Display the Profiling Manager dialog box.

---

**Note**

Fast Models does not use the profiling options.

---

<b>View Profiling ...</b>	Display the Profile Information dialog box.
---------------------------	---

---

**Note**

Fast Models does not use the profiling options.

---

<b>Save Model State ...</b>	Save the current model state. If reloaded, simulation continues from the point where the model state was saved.
<b>Restore Model State ...</b>	Reload a previously saved model state.
<b>Load Debug Info for Module</b>	Load debug information for the module.
<b>Set Parameters</b>	Set parameter values for the model.
<b>Select Targets</b>	Select the execution target within the model.

## Layout menu

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

<b>Layout Control Window</b>	Display the Window to set layout options such as tiling.
------------------------------	--

<b>Load Layout ...</b>	Load a previously saved window layout.
<b>Save Layout ...</b>	Save the current layout. Model state is not saved.
<b>Load Recent Layout</b>	Use a recently used window layout.
<b>Restore Default Layout</b>	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:

<b>New View</b>	Display a new debug view.
<b>Hide</b>	Hide an existing debug view.
<b>Show</b>	Display view that was most recently hidden.
<b>Show All</b>	Displays all previously hidden views.
<b>Close</b>	Close the window that currently has focus.
<b>Arrange Horizontally</b>	Tile all view windows horizontally.
<b>Arrange Vertically</b>	Tile all view windows vertically.
<b>Move</b>	Move a view to the new position specified on the submenu.
<b>Docked Views</b>	Dock or undock the views listed on the submenu.

### Help menu

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

<b>Help ...</b>	Opens this book in Adobe Acrobat Reader.
<b>About ...</b>	Displays the standard About dialog box displaying version and license information.
<b>About Model ...</b>	Opens the text file that contains the release notes.

## 2.2.3 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.

## 2.2.4 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.

The target location 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.

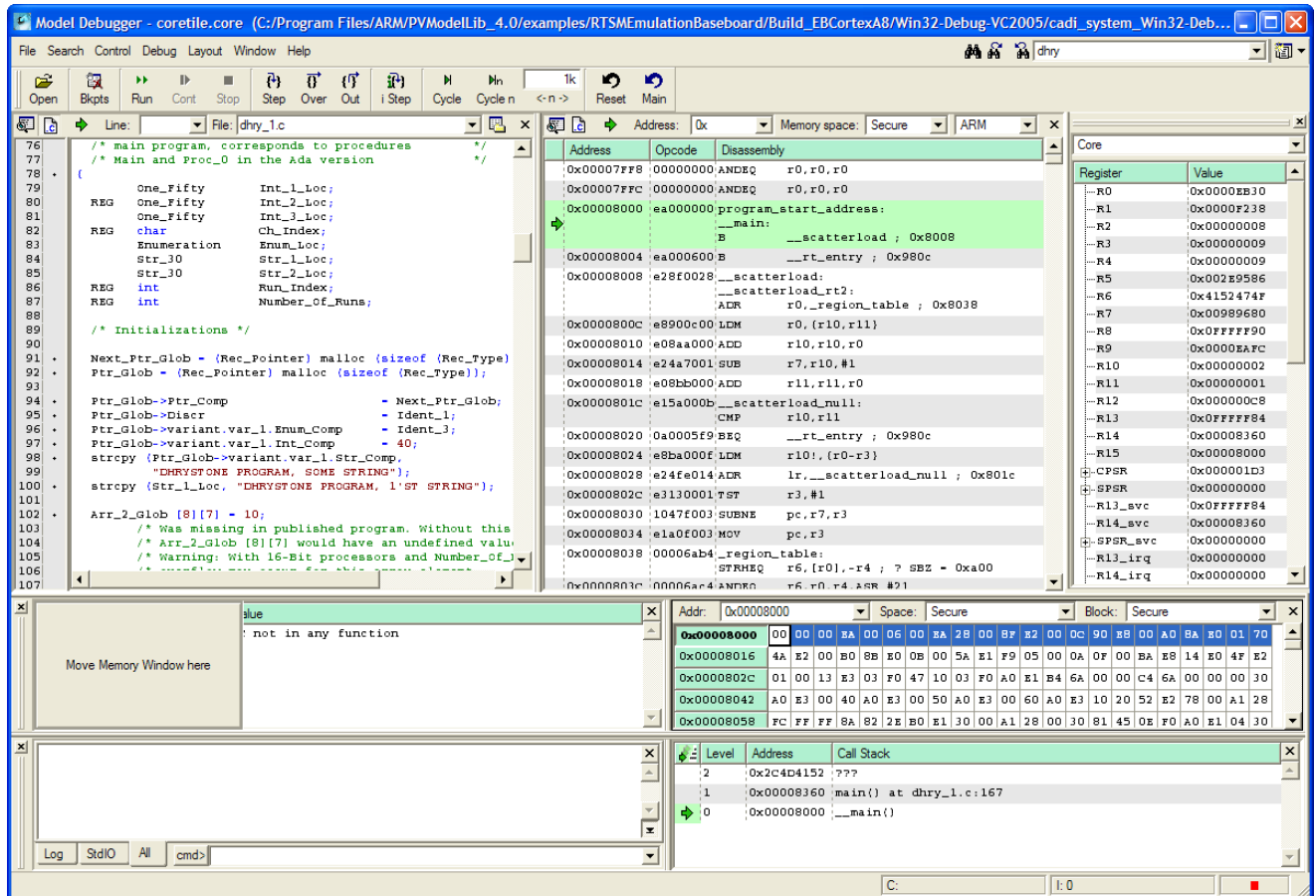


Figure 2-12 Drag-and-drop of debug views, while moving the Memory window

Press **Ctrl+F9** to copy the window. This effectively duplicates the existing view. The location for the duplicate view 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.

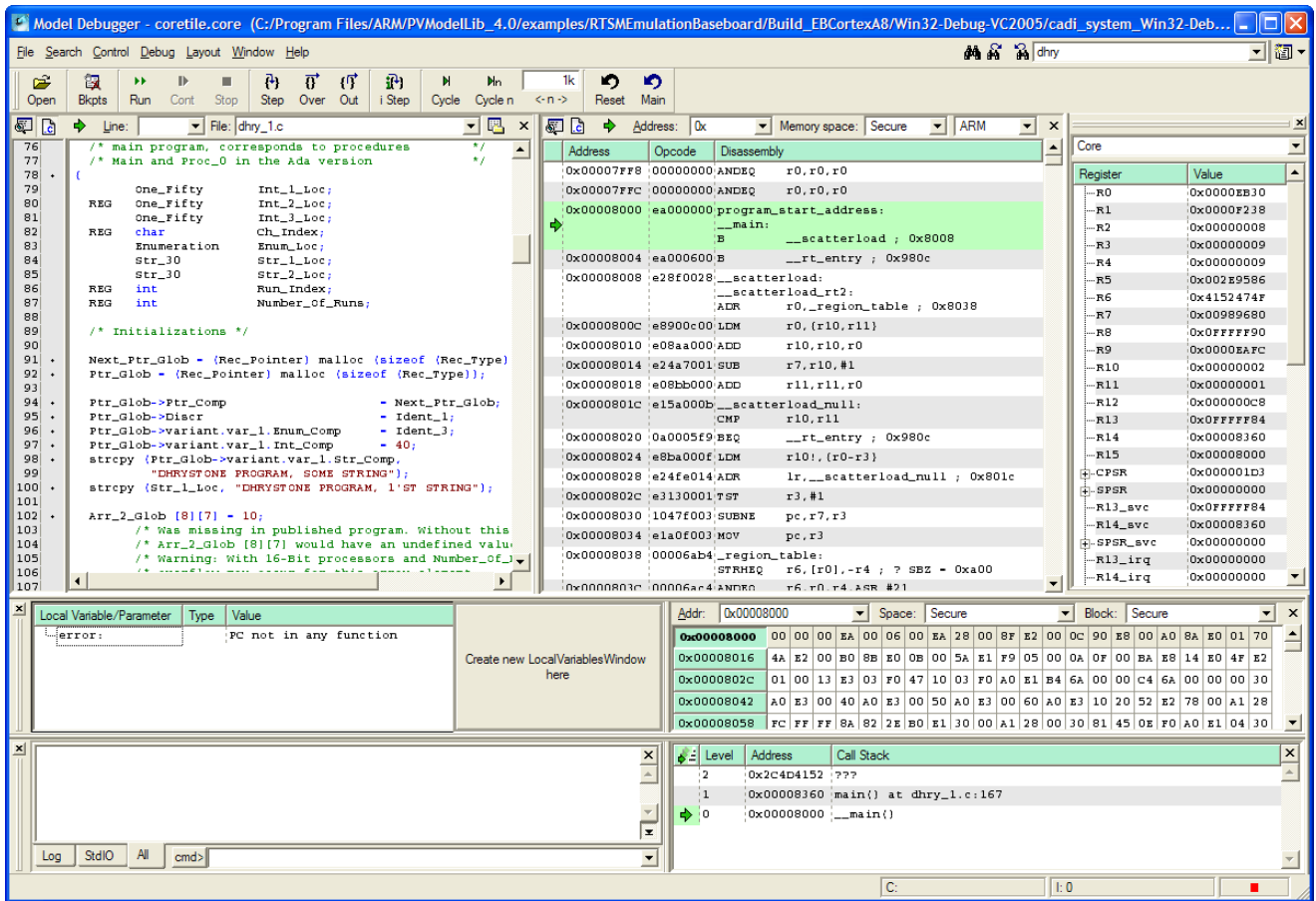


Figure 2-13 Duplicating a register view

### 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 take several moves to force the window to the required location.

## 2.2.5 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. Click on an entry to change focus to the selected window.

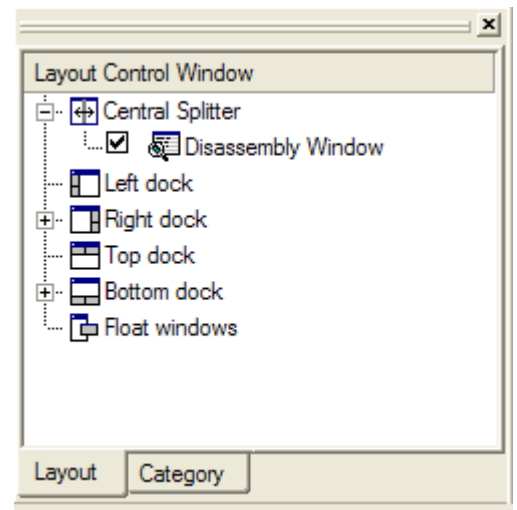


Figure 2-14 Layout Control window

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

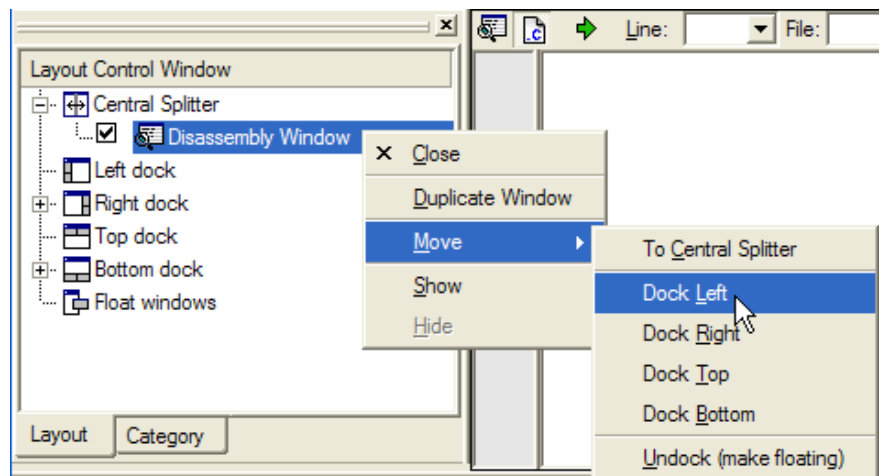


Figure 2-15 Layout Control context menu

### 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.

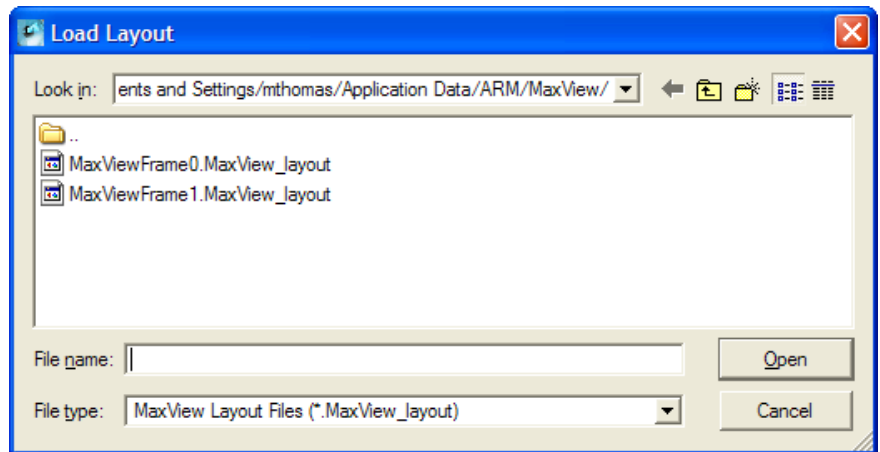


Figure 2-16 Load Layout dialog box

**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.

## 2.2.6 Opening new debug views

Open a new window by either:

- selecting **New View** from the **Window** menu and selecting the required type of debug view.
- clicking on the **View** icon located at the right of the menu bar and selecting a view from the list.

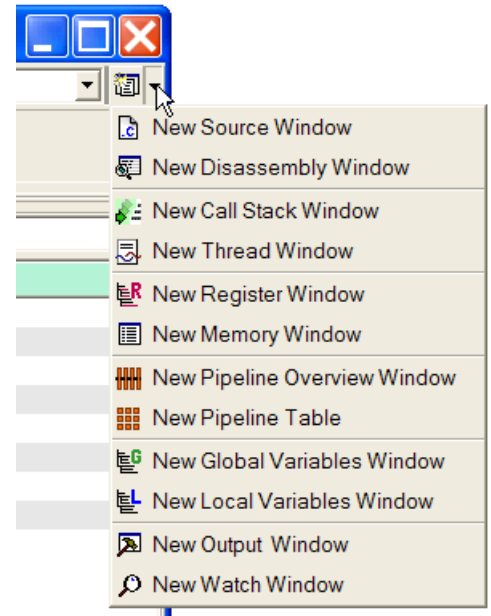


Figure 2-17 Icons for selecting a new debug view

## 2.2.7 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.

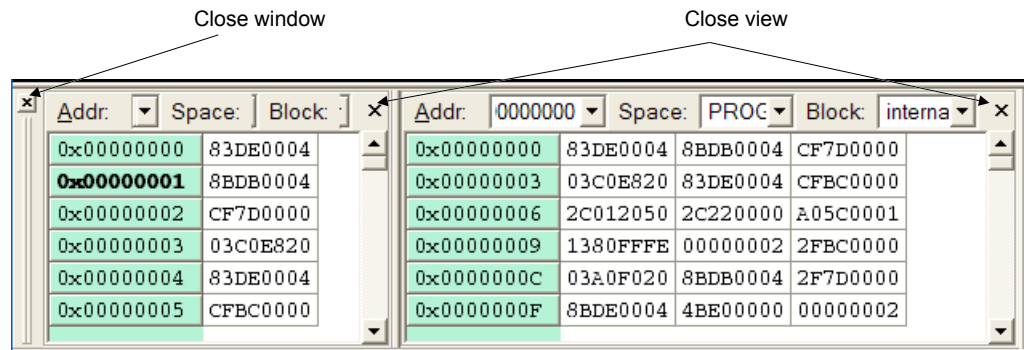


Figure 2-18 Closing windows or individual debug views

## 2.2.8 Output window

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

- Log** Debugger messages, such as errors and warnings.
- StdIO** Output from the target model.
- All** An interleaved view for both the **Log** and **StdIO** categories.

The MxScript 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.

### **Related Information**

*[MxScript for Fast Models Reference Manual.](#)*

### **Related References**

*[Preferences dialog box on page 2-72.](#)*

*[Saving the window layout on page 2-37.](#)*

## 2.3 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.

It contains the following sections:

- [Source view on page 2-42.](#)
- [Disassembly view on page 2-47.](#)
- [Call Stack on page 2-49.](#)

### 2.3.1 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.

The button with the green arrow 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 2-19 Arrow button for scrolling code**

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 filled red circle is displayed next to the line to indicate that a breakpoint has been set.

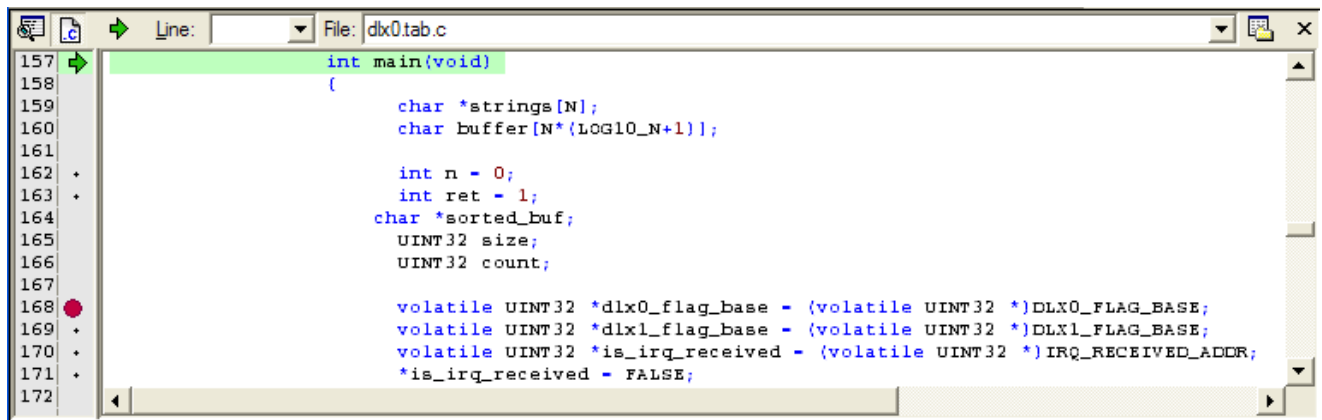


Figure 2-20 Source view

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 box.

### Context menu for Source view

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

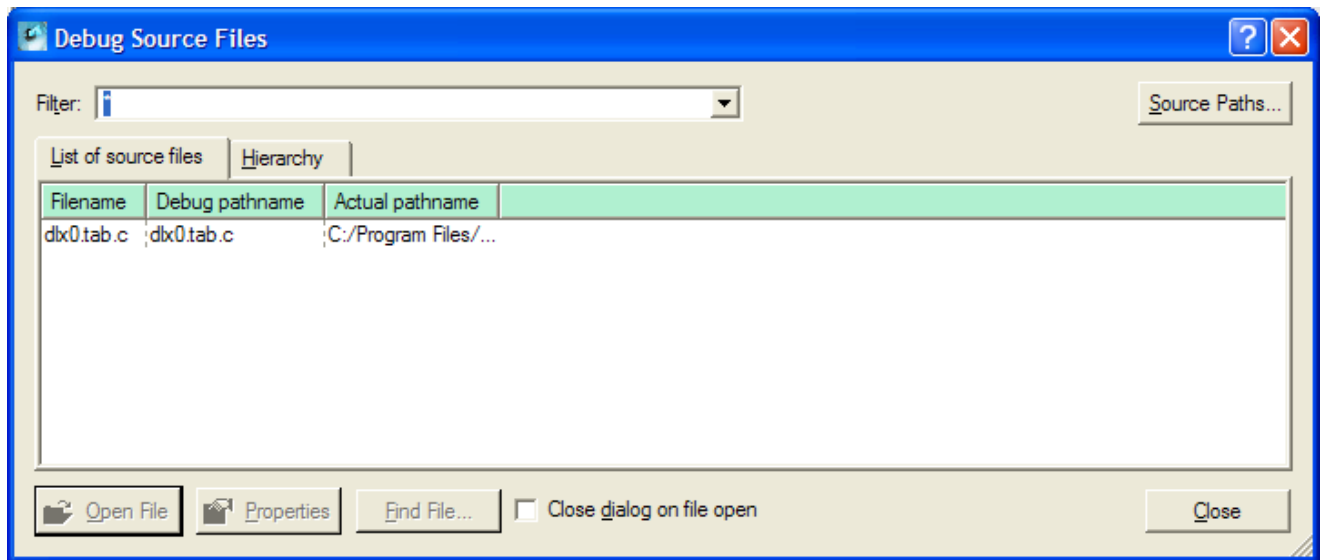
<b>Insert Breakpoint</b>	Insert a breakpoint at the selected location.
<b>Enable Breakpoint</b>	Enable the breakpoint at the selected location.
<b>Breakpoint Properties</b>	If a breakpoint is present on the selected instruction, selecting this option displays the Breakpoint properties dialog box.
<b>Run to here</b>	Run to the selected instruction
<b>Word wrap</b>	Wrap text to fit inside window
<b>File properties</b>	Display the filename and path for the currently displayed file.

### Debug Source Files dialog box

 The Debug Source Files dialog box lets you locate source files required for debugging an application. To open the dialog box, 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 2-21 Debug Source Files dialog box**

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

<b>Filename</b>	This column contains a list of files referred to by the debugged application. This column is not shown in <b>Hierarchy</b> view.
<b>Debug pathname</b>	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.
<b>Actual pathname</b>	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 <b>Open File</b> . The File Open dialog box 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 box and navigate to the directory containing the source.

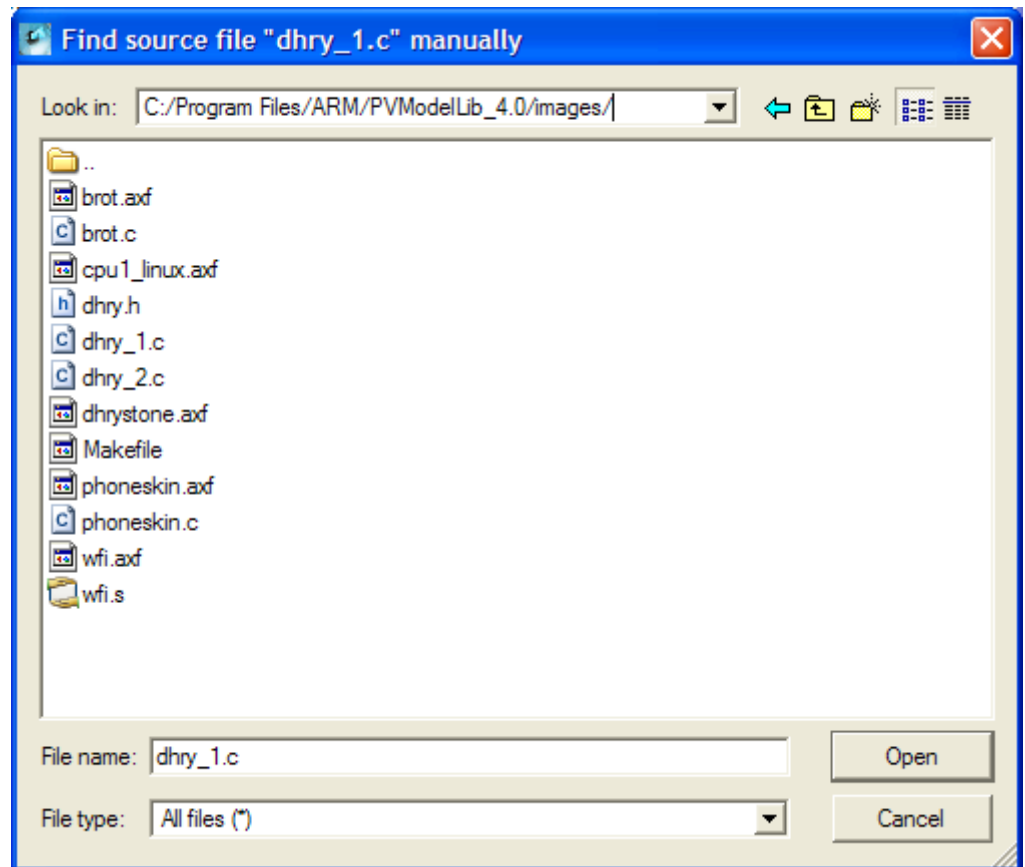


Figure 2-22 Find Source File dialog box

Click on **Properties** to display the File Properties dialog box for the selected file. You can also use the **Find File** button on the File Properties dialog box to locate the file.

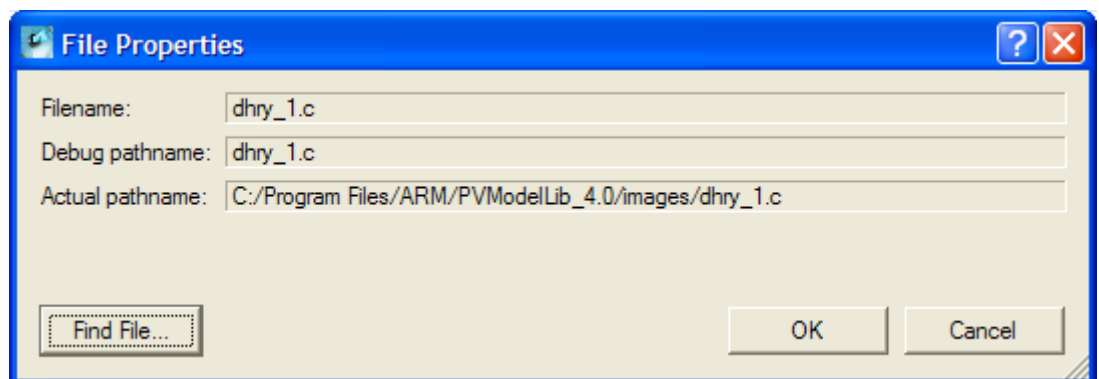


Figure 2-23 Source File Properties dialog box

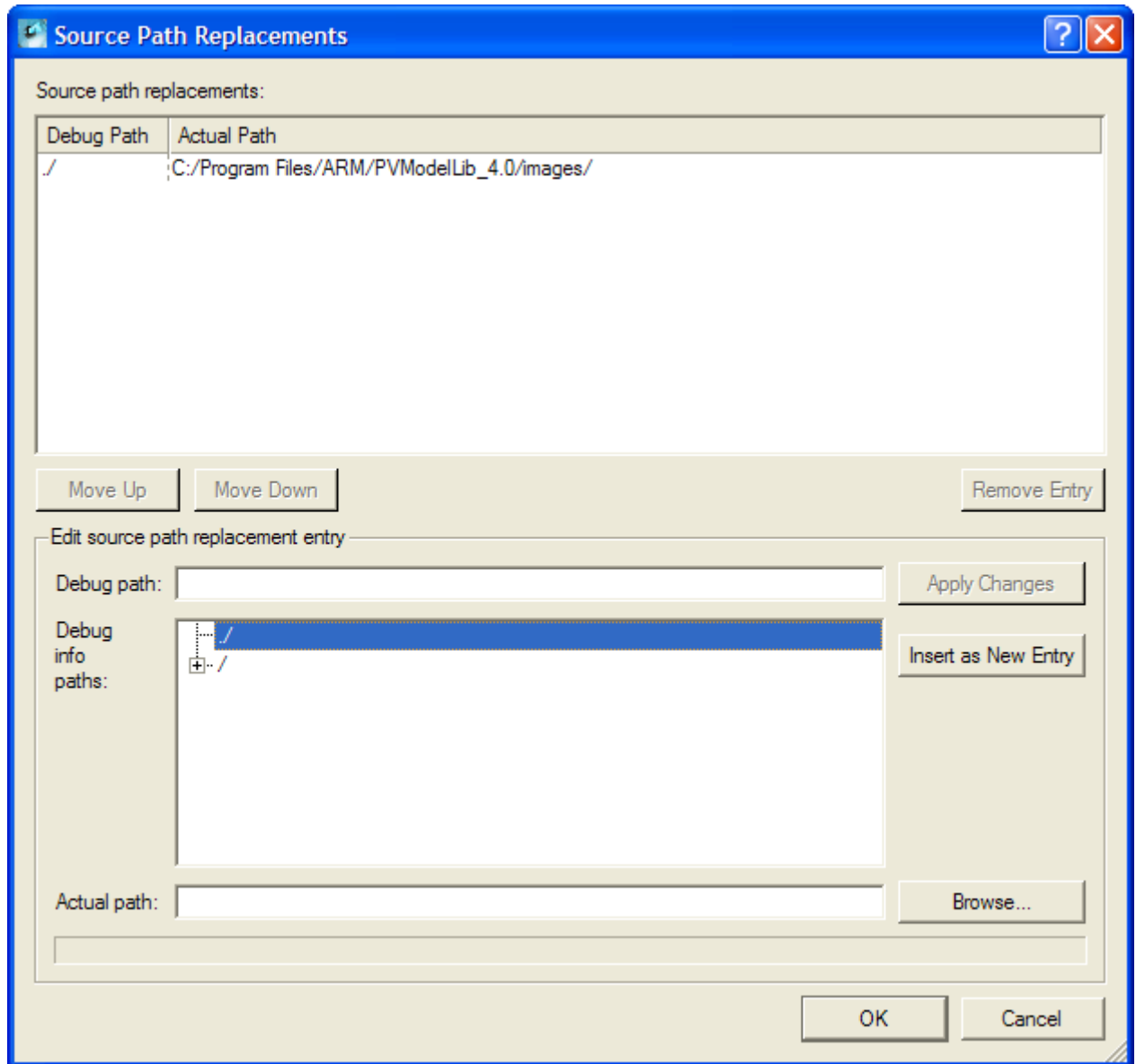
Model Debugger has an automatic mechanism to add replacement paths that are invoked every time you are prompted to find a source file. If the source file is found, an automatic source path replacement is calculated.

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 box. Use this dialog box 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 2-24 Source Path Replacement dialog box**

Existing source file replacements are displayed in the top part of the Source Path Replacement dialog box. 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 box.

In the lower part of the Source Path Replacement dialog box, 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 a 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 box. The active window is surrounded by a thin black frame. Click **Find** on the **Search** menu to open the Find dialog box.

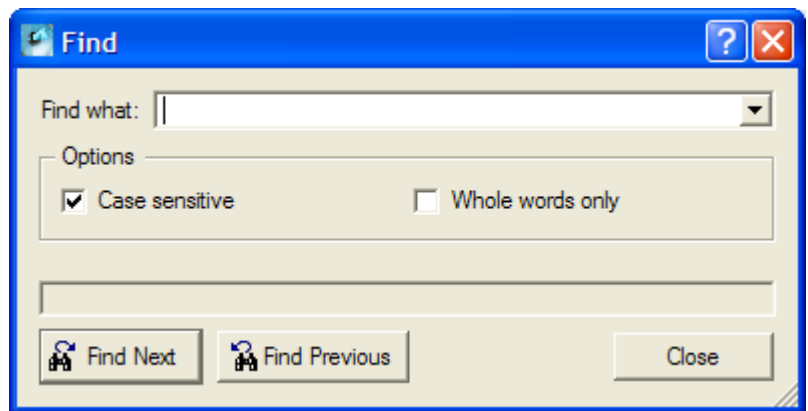


Figure 2-25 Find dialog box

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. Re-use previous search terms by clicking on the drop-down arrow on the right of the text entry box.

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

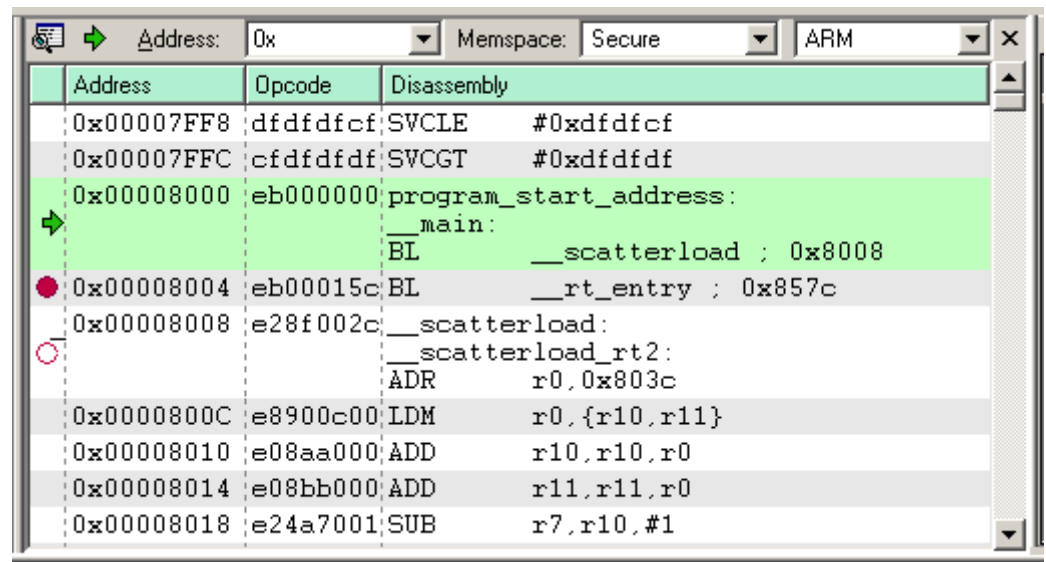
### 2.3.2 Disassembly view

The Disassembly view provides four columns for breakpoints and PC indicator, address, opcode, and disassembly string.

If your target model has TrustZone® support, disassembly breakpoint from all worlds are shown in the first column. The filled red circles indicate a breakpoint in the world currently shown in the disassembly view, and unfilled red circles indicate breakpoints in other worlds.

The green arrow indicates the actual position of the PC.

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.



Address	Opcode	Disassembly
0x00007FF8	dfdfdfcf	SVCLE #0xdfdfcf
0x00007FFC	cfdfdfdf	SVCGT #0xdfdfdf
0x00008000	eb000000	program_start_address: __main: BL __scatterload ; 0x8008
0x00008004	eb00015c	BL __rt_entry ; 0x857c
0x00008008	e28f002c	__scatterload: __scatterload_rt2: ADR r0, 0x803c
0x0000800C	e8900c00	LDM r0, {r10, r11}
0x00008010	e08aa000	ADD r10, r10, r0
0x00008014	e08bb000	ADD r11, r11, r0
0x00008018	e24a7001	SUB r7, r10, #1

Figure 2-26 Disassembly view

The Disassembly view title bar has the following controls:

- Address:** Enter a start address to display the code from.
- Memory space:** Select **Secure** (TrustZone) or **Normal** memory space, if applicable for the processor architecture.
- Architecture** 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.



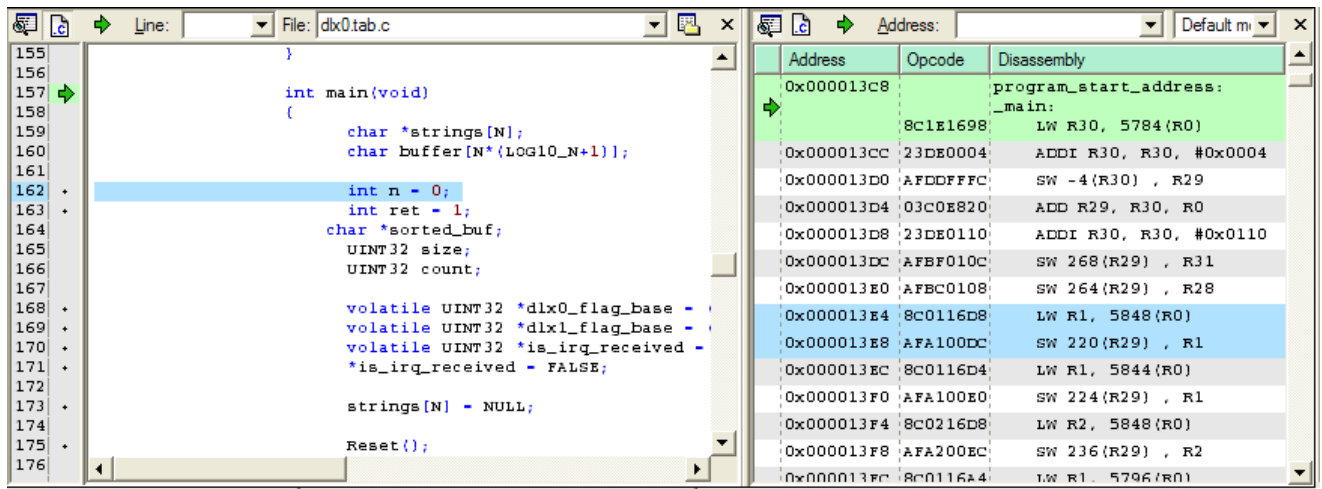


Figure 2-27 Matching source and disassembly

### Context menu for Disassembly view

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

<b>Insert/Remove Breakpoint</b>	Insert/Remove a breakpoint on the selected location only in the current shown memory space (TrustZone world). The same can be achieved with a double click in the first column.
<b>Insert/Remove Breakpoint into /from all Program Memories</b>	Insert /Remove a breakpoint on the selected location in all program memory spaces.
<b>Enable/Disable Breakpoint</b>	Enable/Disable the breakpoint at the selected location.
<b>Breakpoint Properties</b>	If a breakpoint is present on the selected location, selecting this option displays the Breakpoint properties dialog box.
<b>Show memory</b>	Display a dialog box to select a memory space and update the Memory view to display the memory contents at the address specified corresponding to the instruction location.
<b>Run to here</b>	Step the code until the selected location is reached.

### 2.3.3 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.

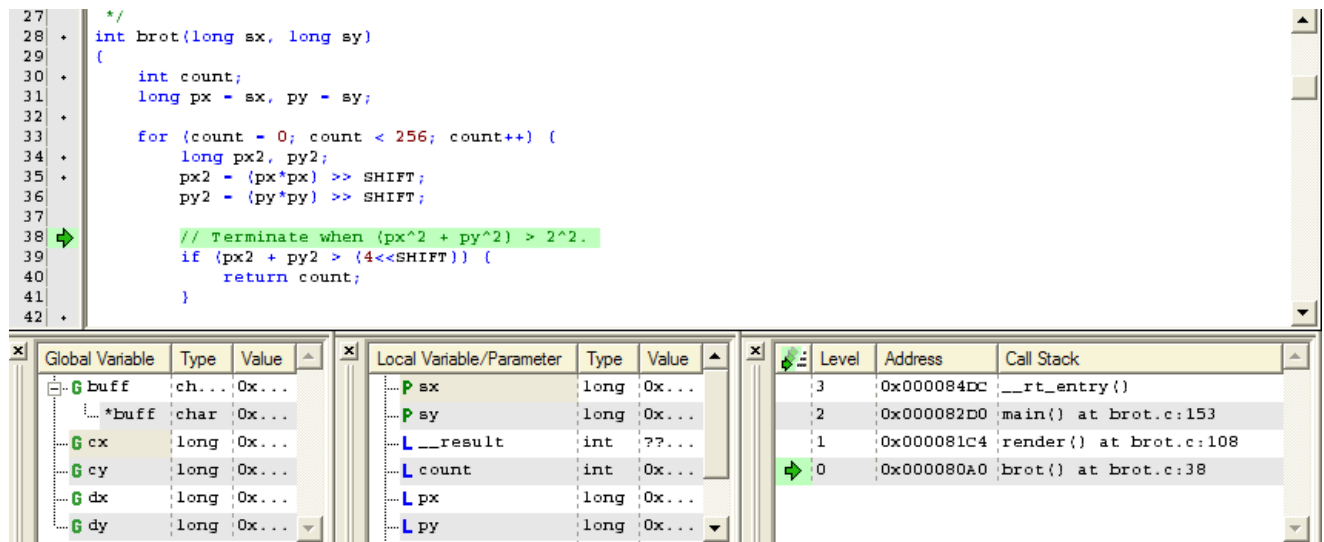


Figure 2-28 Call Stack view

#### 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.

## 2.4 Debug views for registers and memory

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

It contains the following sections:

- [Register views on page 2-51.](#)
- [Memory on page 2-53.](#)
- [Variables on page 2-55.](#)

### 2.4.1 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.

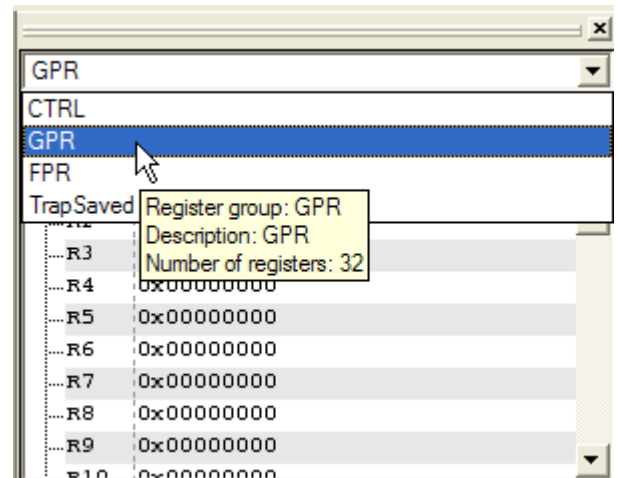


Figure 2-29 Select register group

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

- Use the context menu in the register view and select **Show Previous Values**.

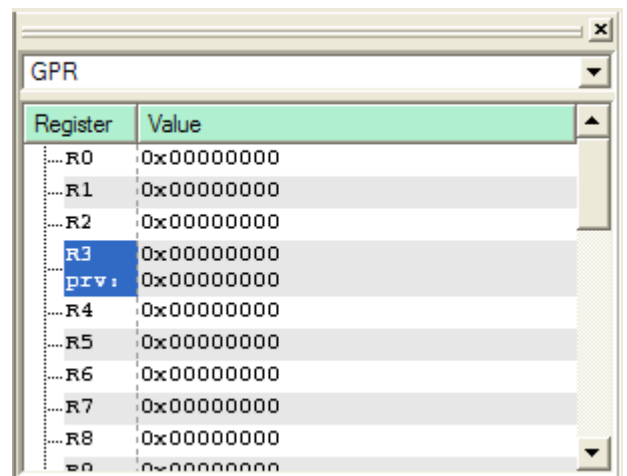


Figure 2-30 Register view showing current and previous contents

- Place the cursor over the respective register. The buffered state is updated every time the model execution stops.

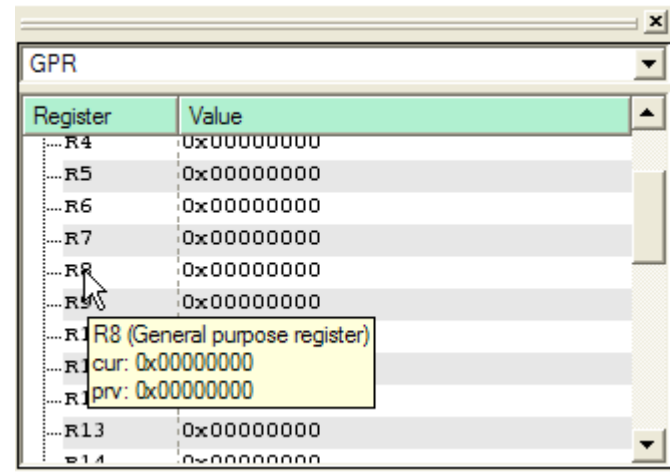


Figure 2-31 Register view contents at cursor

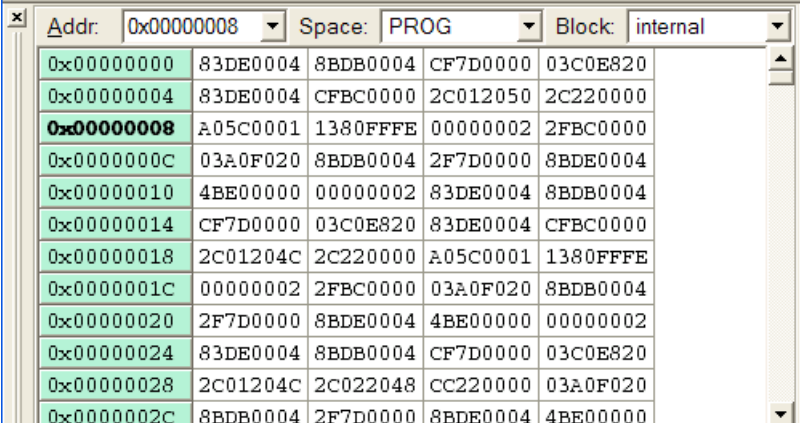
### Context menu for Register view

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

<b>Copy</b>	Copy the contents of the selected register.
<b>Add to Watch</b>	Add the selected register to the Watch view.
<b>Insert Breakpoint</b>	Insert a breakpoint on the selected register.
<b>Enable Breakpoint</b>	Enable the breakpoint at the selected register.
<b>Breakpoint Properties</b>	If a breakpoint is present on the selected register, selecting this option displays the Breakpoint properties dialog box.
<b>Edit Value</b>	Edit the contents for the selected register.
<b>Select and show memory at <i>nnn</i></b>	Display a dialog box to select a memory space and update the Memory view to display the memory contents at the address specified by the register contents.
<b>Show memory at <i>nnn</i></b>	Update the Memory view to display the memory contents at the address specified by the register contents.
<b>Format</b>	Choose the number base to use to display the register contents. The options are <b>Default Format</b> , <b>Unsigned Decimal</b> , <b>Signed Decimal</b> , <b>Hexadecimal</b> , <b>Binary</b> , <b>Float</b> , or <b>ASCII</b> .
<b>Show Previous Value</b>	Display the current value and the previous value for the selected register.
<b>Select All</b>	Select all of the registers in the Register view.

## 2.4.2 Memory

The Memory view displays a range of memory starting from the base address specified in the address field (**Addr:**). Enter base addresses as decimal numbers or, by using the prefix 0x, as hexadecimal numbers. Additional fields allow for selection of the address space (**Space:**) and physical memory block (**Block:**).



Addr:	Space:	Block:
0x00000000	PROG	internal
0x00000000	83DE0004	8BDB0004
0x00000004	83DE0004	CFBC0000
0x00000008	A05C0001	1380FFFE
0x0000000C	03A0F020	8BDB0004
0x00000010	4BE00000	00000002
0x00000014	CF7D0000	03C0E820
0x00000018	2C01204C	2C220000
0x0000001C	00000002	2FBC0000
0x00000020	2F7D0000	8BDE0004
0x00000024	83DE0004	8BDB0004
0x00000028	2C01204C	2C022048
0x0000002C	8BDB0004	2F7D0000

Figure 2-32 Memory view

### Context menu for Memory view

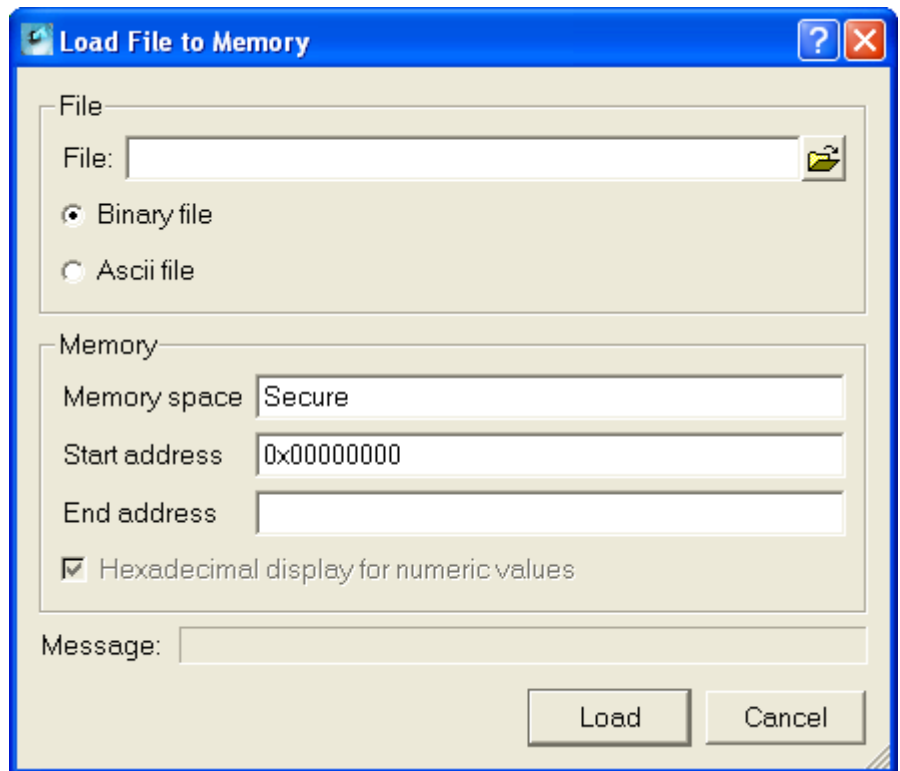
Right-click one of the cells the Memory view to display the context menu. The menu has the following options:

<b>Insert Breakpoint</b>	Insert a breakpoint on the selected memory location.
<b>Enable Breakpoint</b>	Enable the breakpoint at the selected memory location.
<b>Breakpoint Properties</b>	If a breakpoint is present on the selected memory location, selecting this option displays the Breakpoint properties dialog box.
<b>Edit Value</b>	Edit the contents for the selected memory location.
<b>Select and show memory at <i>nnn</i></b>	Display a dialog box 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.
<b>Show memory at <i>nnn</i></b>	Update the Memory view to display the memory contents at the address specified by the contents of the memory location.
<b>Show disassembly at <i>nnn</i></b>	Update the disassembly view to display the disassembly contents at the address specified by the contents of the memory location.
<b>Copy</b>	Copy the contents of the selected memory location.
<b>Add to Watch</b>	Add the selected memory location to the Watch view.
<b>Endian</b>	Select the memory model to use to display memory contents. The options are: <b>Default Endian</b> , <b>Little Endian</b> , and <b>Big Endian</b> .

<b>Format</b>	Choose the number base to use to display the memory contents. The options are <b>Default Format</b> , <b>Unsigned Decimal</b> , <b>Signed Decimal</b> , <b>Hexadecimal</b> , <b>Binary</b> , <b>Float</b> , or <b>ASCII</b> .
<b>Fixed column count</b>	Display a fixed number of memory values per row. The number to be displayed is determined by the width of the memory window.
<b>Increment column count</b>	Increment the number of memory values to display per row.
<b>Decrement column count</b>	Decrement the number of memory values to display per row.
<b>Increment current address</b>	Increment the start address used for each memory row.
<b>Decrement current address</b>	Decrement the start address used for each memory row.
<b>Increment MAU per cell</b>	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.
<b>Decrement MAU per cell</b>	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.
<b>Load File to Memory ...</b>	Open Load File to Memory dialog box to load a binary or ascii file into memory.
<b>Save Memory in a File ...</b>	Open Save Memory in a File dialog box to save the contents of memory in a binary or ascii file.
<b>Memory Display Options</b>	Display the Memory Display Options dialog box to enable setting column count, view format, endian mode, and MAU per cell.

#### Load File to Memory dialog box/Store Memory in a File

Use these dialog boxes to load or store the memory contents of the target model. They have the same look for loading and storing.



**Figure 2-33 Load File to Memory dialog box**

Enter the file name into the top field of the dialog box. You can use the button next to it to browse for the file. Select if a binary or ascii file is to be loaded or stored. The Memory Space and Start Address fields are filled automatically from the memory view where you opened the dialog box. You can change the values. Put an end address into the bottom field. If you do not enter a value here - only when loading a file - the max address of the memory is used.

Click Store/Load button for store/load of the file. If any problems occur a message appears in the Message field.

### 2.4.3 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, with their type and value.

- A blue L before the variable name indicates a local variable.
- A green leading P indicates a parameter.

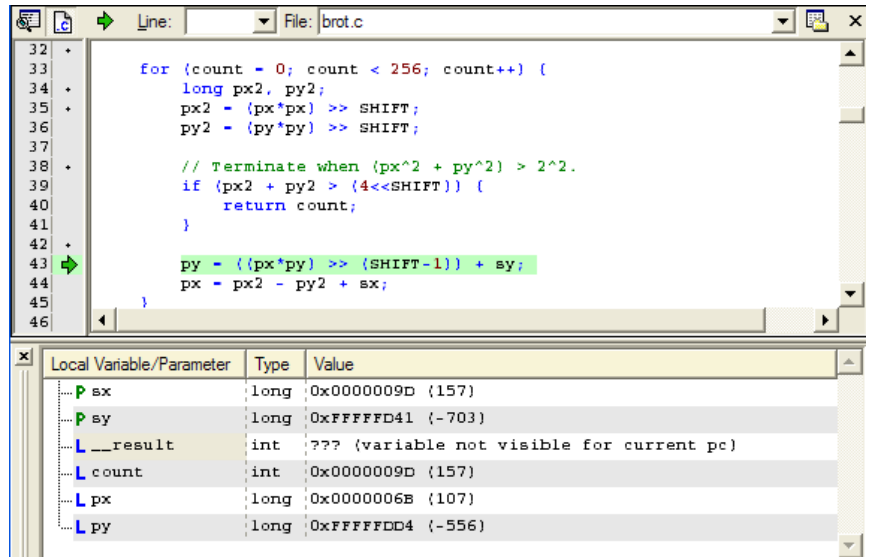


Figure 2-34 Local Variable view

### Global Variables window

This window shows the global variables with their types and values. They are marked by a green G.

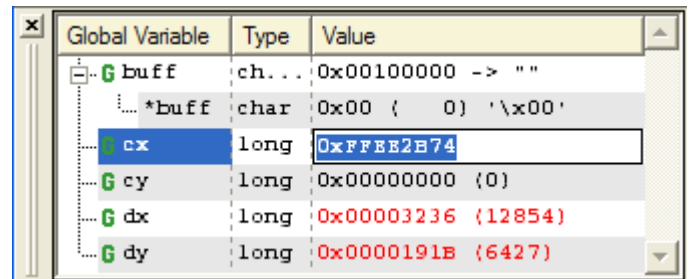


Figure 2-35 Global Variable view

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 box 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:

<b>Copy</b>	Copy the contents of the selected variable.
<b>Add to Watch</b>	Add the selected variable to the Watch view.
<b>Insert Breakpoint</b>	Insert a breakpoint on the selected variable.
<b>Enable Breakpoint</b>	Enable the breakpoint at the selected variable.
<b>Breakpoint Properties</b>	If a breakpoint is present on the selected variable, selecting this option displays the Breakpoint properties dialog box.
<b>Edit Value</b>	Edit the contents of the selected variable.
<b>Show memory</b>	Display a dialog box 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.
<b>Show Previous Value</b>	Display the current value and the previous value for the selected variable.
<b>Select All</b>	Select all of the variables in the variable view.
<b>Load Debug Info for Module</b>	Load debug information for the module that contains the selected variable.

## 2.5 Debug views for pipelines

Model Debugger provides options for viewing the pipeline:

- The Pipeline Overview window.
- The Pipeline Table.

———— **Note** ————

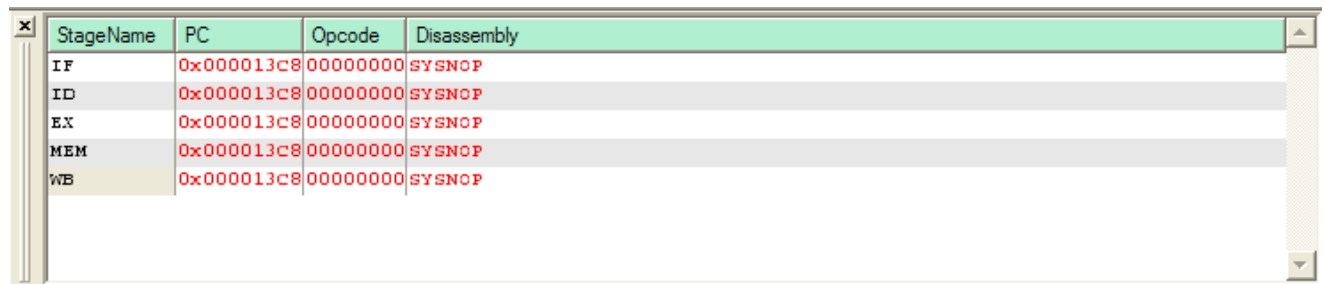
Pipeline views are only available if your model supports them. If the pipeline icons are gray, not orange, then you cannot view pipeline information.

It contains the following sections:

- [Pipeline Overview window on page 2-58.](#)
- [Pipeline Table window on page 2-58.](#)

### 2.5.1 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.



StageName	PC	Opcode	Disassembly
IF	0x000013c8	00000000	SYSNOP
ID	0x000013c8	00000000	SYSNOP
EX	0x000013c8	00000000	SYSNOP
MEM	0x000013c8	00000000	SYSNOP
WB	0x000013c8	00000000	SYSNOP

**Figure 2-36 Pipeline Overview window**

### 2.5.2 Pipeline Table window

The Pipeline Table 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.

IF		ID		EX		MEM		WB	
PC	0x000013c8	PC	0x000013c8	PC	0x000013c8	PC	0x000013c8	PC	0x000013c8
CON	3	CON	3	CON	3	CON	3	CON	3
opc	00000000	opc	00000000	opc	00000000	opc	00000000	opc	00000000
dis	SYSNOP	dis	SYSNOP	dis	SYSNOP	dis	SYSNOP	dis	SYSNOP
it	0x00000000	it	0x00000000	it	0x00000000	it	0x00000000	it	0x00000000
a	0x00000000	a	0x00000000	a	0x00000000	a	0x00000000	a	0x00000000
b	0x00000000	b	0x00000000	b	0x00000000	b	0x00000000	b	0x00000000
d	0x00000000	d	0x00000000	d	0x00000000	d	0x00000000	d	0x00000000
i	0x00000000	i	0x00000000	i	0x00000000	i	0x00000000	i	0x00000000
Ra	0x00000000	Ra	0x00000000	Ra	0x00000000	Ra	0x00000000	Ra	0x00000000
Rb	0x00000000	Rb	0x00000000	Rb	0x00000000	Rb	0x00000000	Rb	0x00000000
Rd	0x00000000	Rd	0x00000000	Rd	0x00000000	Rd	0x00000000	Rd	0x00000000
ALU	0x00000000	ALU	0x00000000	ALU	0x00000000	ALU	0x00000000	ALU	0x00000000
LMD	0x00000000	LMD	0x00000000	LMD	0x00000000	LMD	0x00000000	LMD	0x00000000
con	0x00	con	0x00	con	0x00	con	0x00	con	0x00
mem	0x00000000	mem	0x00000000	mem	0x00000000	mem	0x00000000	mem	0x00000000
mem	0x00	mem	0x00	mem	0x00	mem	0x00	mem	0x00
isc	0x00	isc	0x00	isc	0x00	isc	0x00	isc	0x00

Figure 2-37 Pipeline Table window

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.

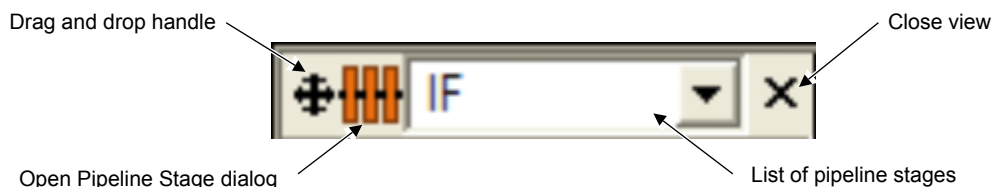


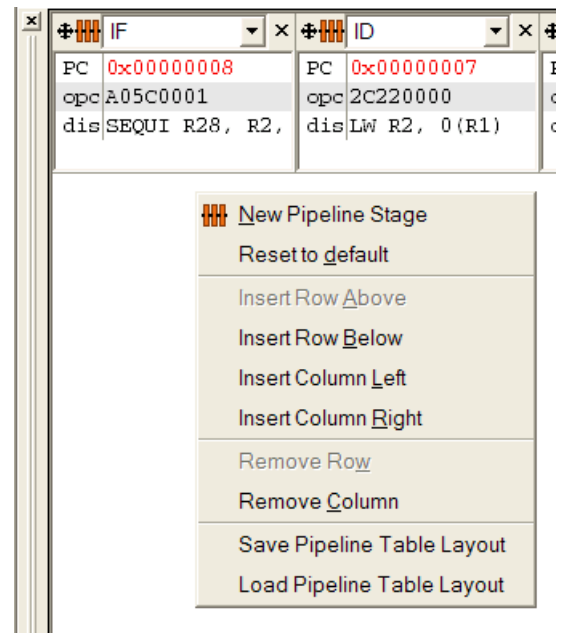
Figure 2-38 Pipeline Table icons

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 on empty space in the table or in an empty cell to open the context menu.



**Figure 2-39 Pipeline Table context menu**

The context menu has the following entries:

<b>New Pipeline Stage</b>	Select to create a new stage and add it to the view.
<b>Reset to default</b>	Select to use the default layout for the Pipeline Table view.
<b>Insert Row Above/Insert Row Below</b>	Select to insert a new row above or below the current cell.
<b>Insert Column Left/Insert Column Right</b>	Select to insert a new column to the left or right of the current cell.
<b>Remove Row/Remove Column</b>	Select to remove the row or column that includes the current cell.
<b>Save Pipeline Table Layout</b>	Select to save the current layout to file.
<b>Load Pipeline Table Layout</b>	Select to load a previously saved layout and use that configuration in the Pipeline Table view.

### Pipeline Stage Properties dialog box

Click the orange icon to the right of the cross to open the Pipeline Stage Properties dialog box. Use this to customize the lists in the Pipeline table. The combo box 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.

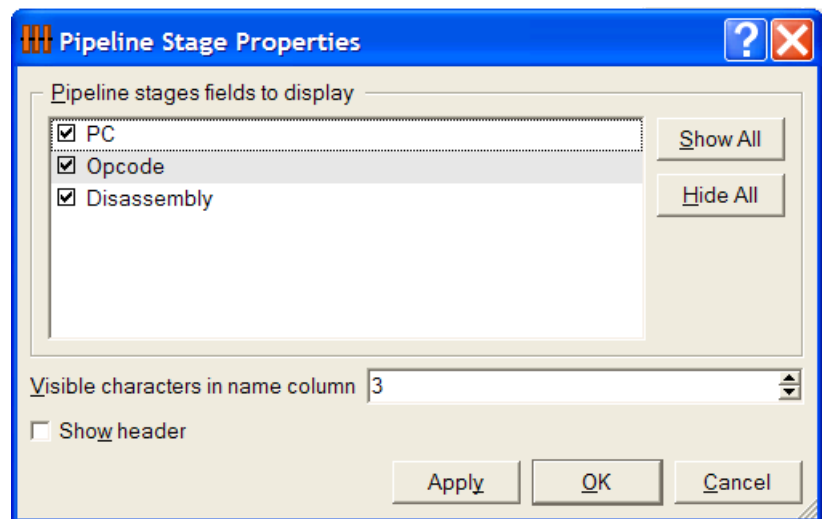


Figure 2-40 Pipeline Stage Properties dialog box

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.

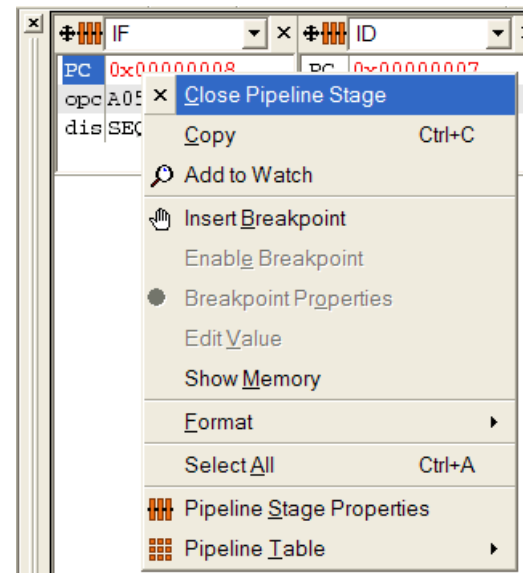


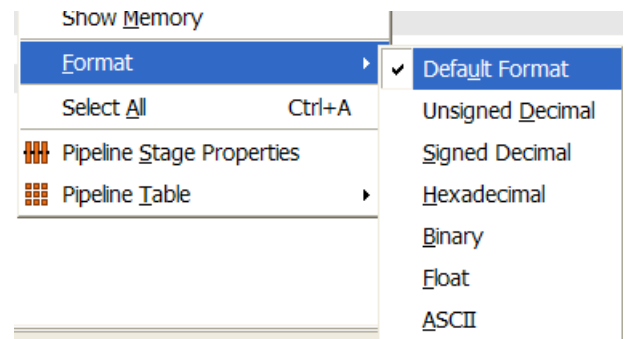
Figure 2-41 Pipeline view context menu

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.

<b>Add to Watch</b>	Select to add the pipeline field to the Watch window.
<b>Remove Breakpoint/ Insert Breakpoint</b>	<p>The text displayed depends on whether the selected field already has a breakpoint:</p> <ul style="list-style-type: none"> <li>• If a breakpoint is present, select <b>Remove Breakpoint</b> to delete it.</li> <li>• If a breakpoint is not present, select <b>Insert Breakpoint</b> to add a breakpoint.</li> </ul>
<b>Enable Breakpoint/ Disable Breakpoint</b>	<p>If a breakpoint is present:</p> <ul style="list-style-type: none"> <li>• Select <b>Enable Breakpoint</b> to enable it</li> <li>• Select <b>Disable Breakpoint</b> to retain the breakpoint, but disable it.</li> </ul>
<b>Breakpoint Properties</b>	<p>Select to open a dialog box to view and set the details and conditions for a particular breakpoint.</p> <p>This dialog box is also available from the Breakpoint Manager dialog box.</p>
<b>Edit Value</b>	Select to open the chosen field for inline edit.
<b>Show Memory</b>	Select to mark the memory address in the Memory Window.
<b>Format</b>	Select to open the submenu. This menu lets you choose the format for number display.



**Figure 2-42 Submenu for display format**

<b>Select All</b>	Selects all fields in the list.
<b>Pipeline Stage Properties</b>	Select to open the Pipeline Stage Properties dialog box to change the contents of the Pipeline Stage view.
<b>Pipeline Table</b>	Select to display a submenu for the Pipeline Table.

## Related References

*Breakpoint Manager dialog box on page 2-68.*

## 2.6 Watch window and Expression Evaluator

The Expression Evaluator is located in the Watch window. To display the window, select **Window > New View > New Watch Window**.

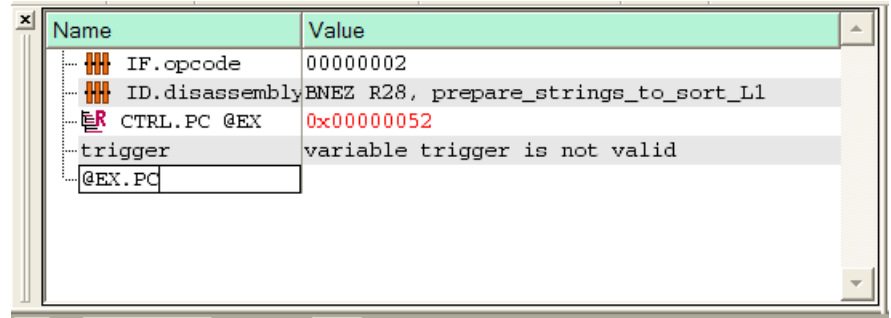


Figure 2-43 Watch window

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 way 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 required 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.
- Hexadecimals have a leading 0x.
- Literals with no prefix are interpreted as decimals.

**Floating-point values** Floating point values can have decimal and scientific representation. Enter floating point values in decimal representation (123.456) or in scientific representation with positive or negative exponent (1.23456e2).

Combine resources, variables and literals in the target 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. 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 2-2 Operator precedence**

Precedence	Operators	Associativity
1	[] -> .	Left to right
2	! ~ + - * & (unary) (cast) sizeof	Right to left
3	* (binary) / %	Left to right
4	+ - (binary)	Left to right
5	<< >>	Left to right
6	< <= > >=	Left to right
7	== !=	Left to right
8	& (binary)	Left to right
9	^	Left to right
10		Left to right



**Table 2-2 Operator precedence (continued)**

Precedence	Operators	Associativity
11	&&	Left to right
12		Left to right
13	?:	Left to right

It contains the following sections:

- *Context menu for Watch window on page 2-65.*

### 2.6.1 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:

<b>Paste</b>	Insert a copied memory, register, or variable into the Watch window.
<b>Copy</b>	Copy an item and its value from the Watch window.
<b>Insert Breakpoint</b>	Insert a breakpoint on the selected watched item.
<b>Enable Breakpoint</b>	Enable the breakpoint at the selected watched item.
<b>Breakpoint Properties</b>	If a breakpoint is present on the selected watched item, selecting this option displays the Breakpoint properties dialog box.
<b>Edit Value</b>	Edit the contents for the selected watched item.
<b>Select and show memory at <i>nnn</i></b>	Display a dialog box 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.
<b>Show memory at <i>nnn</i></b>	Update the Memory view to display the memory contents at the address specified by the contents of the watched item.
<b>Format</b>	Choose the number base to use to display the watched item. The options are <b>Default Format</b> , <b>Unsigned Decimal</b> , <b>Signed Decimal</b> , <b>Hexadecimal</b> , <b>Binary</b> , <b>Float</b> , or <b>ASCII</b> .
<b>Increment number of bytes</b>	Increment the number of memory addresses to display.
<b>Decrement number of bytes</b>	Decrement the number of memory addresses to display.
<b>Select all</b>	Select all of the watched items.

## 2.7 Setting breakpoints in the debug views

Set breakpoints in the 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.

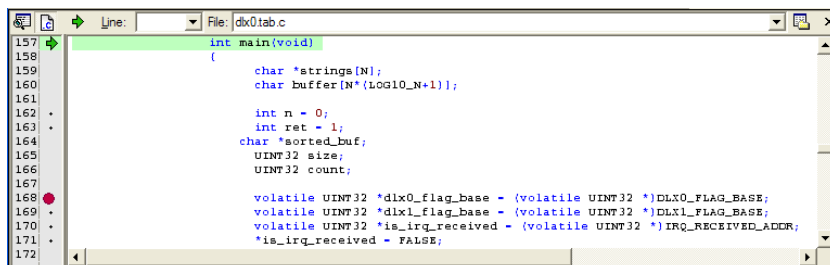


Figure 2-44 Source view breakpoint

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

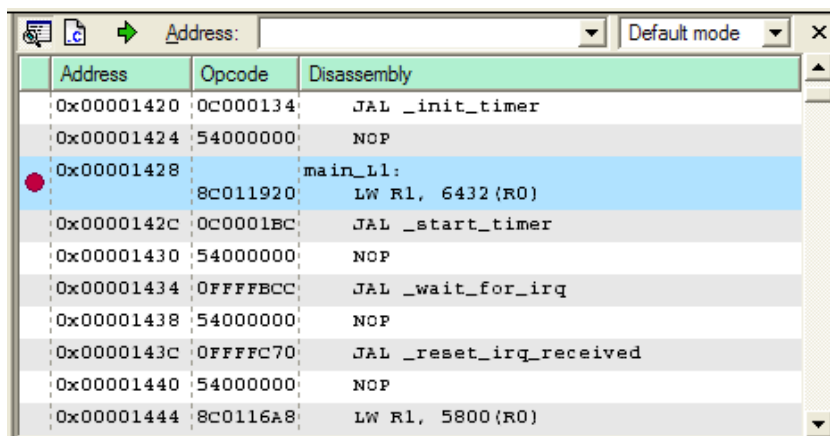


Figure 2-45 Disassembly view breakpoint

## Register view

Double-click on the first column, the register name column, to set breakpoints.

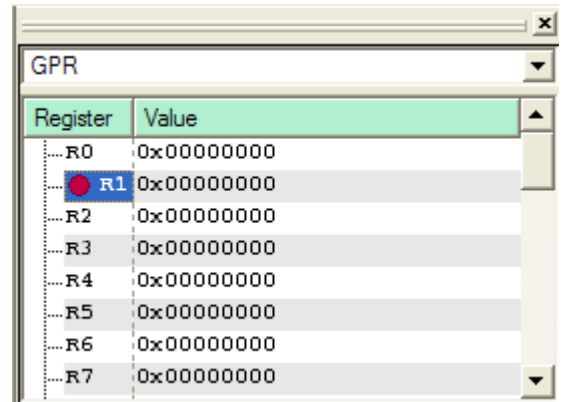


Figure 2-46 Register view breakpoint

## 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.

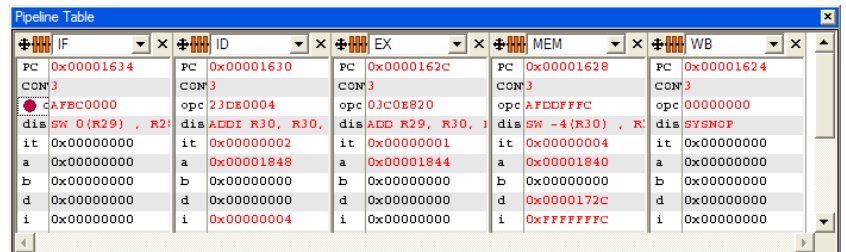


Figure 2-47 Pipeline table breakpoint

**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.

It contains the following sections:

- *Setting conditional breakpoints on page 2-68.*
- *Removing and disabling breakpoints on page 2-68.*
- *Breakpoint Manager dialog box on page 2-68.*
- *Breakpoint Properties dialog box on page 2-69.*

### 2.7.1 Setting conditional breakpoints

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

#### Procedure

1. Setting an unconditional breakpoint.
2. Using the Breakpoint Properties dialog box to set the conditions for the breakpoint.

#### Related References

*Breakpoint Properties dialog box on page 2-69.*

### 2.7.2 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 dialog boxes and menus also permit you to configure your breakpoints.

### 2.7.3 Breakpoint Manager dialog box

All breakpoints can be controlled and maintained through the Breakpoint Manager. This dialog box 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.

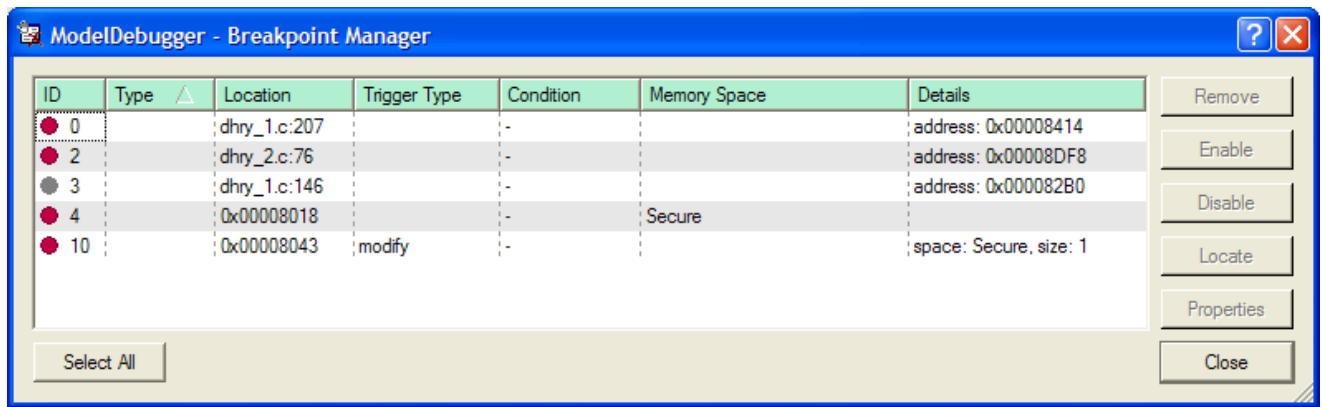
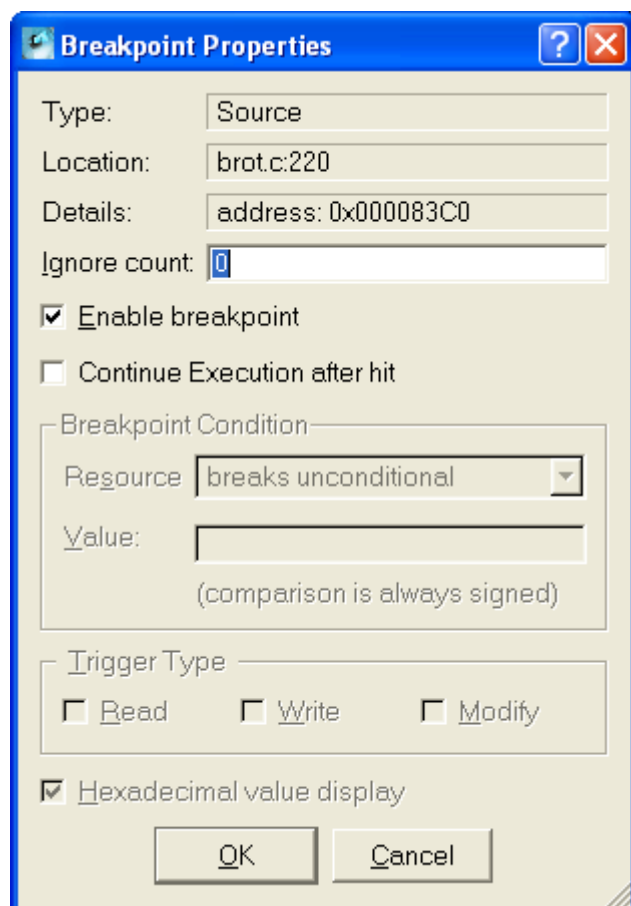


Figure 2-48 Breakpoint Manager dialog box

#### 2.7.4 Breakpoint Properties dialog box

The Breakpoint Properties dialog box can be obtained by right clicking on a breakpoint and then selecting **Properties** from the resulting context menu, and also from the Breakpoint Manager. Use the Breakpoint Properties dialog box, 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.
- Continue Execution after hit** Check this box to enable continue execution after breakpoint hit. If checked the execution of the debugged application does not stop when the breakpoint is being hit.
- 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 is to 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 2-49 Breakpoint Properties dialog box**

## 2.8 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.

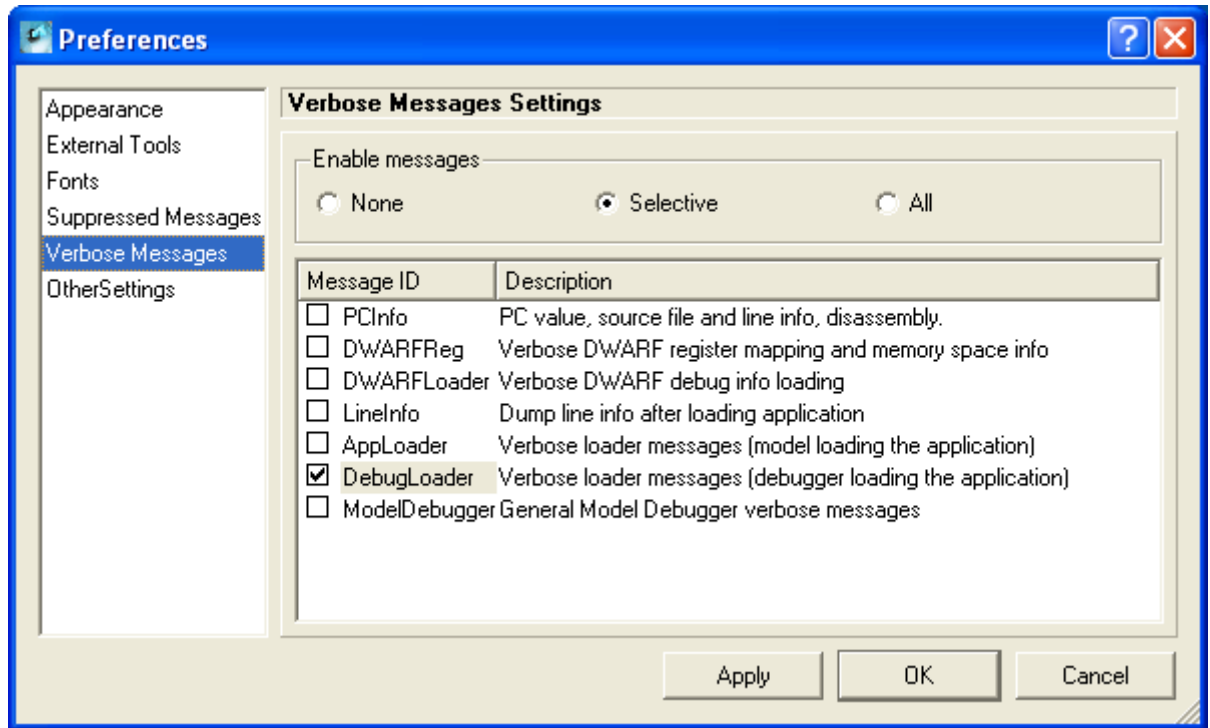
Session files also enable configuring the individual layout of debugger windows for multiprocessor systems. This might be useful, for example, if the project is used with SoC Designer.

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.

## 2.9 Preferences dialog box

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



**Figure 2-50 Preferences dialog box**

Select an entry in the list on the left side of the dialog box 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.



## External Tools

The **use operating system file associations** checkbox is only available on Windows, and is selected by default. This inactivates the external tool edit fields and buttons. To activate these fields, clear the checkbox.

### ————— Note —————

The default external tools are different on Linux.

Use these settings to configure the tools used to display Fast Models documentation. Documentation is accessible by:

- the Model Debugger **Help** menu item. You access the PDFs for the Model Debugger and Fast Model Portfolio books this way.
- The `documentation_file` property in a component property listing. This property might point to a PDF file, a text file, an HTML file, or `http://` link.

You can change the default external tools used to access the documentation. Click on the folder icon to open a browser, or use the drop down list to choose a previously-selected executable.

## Fonts

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

Check the **Fonts depend on \$DISPLAY 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 box 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 processors. The PC is set to the value of the first processor in all processors.

After displaying the dialog box and modifying preferences:

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

## Chapter 3

# Installation and Configuration

This chapter describes how to install and configure a standalone version of Model Debugger. Model Debugger is automatically installed with Fast Models.

It contains the following sections:

- *Linux installation procedure on page 3-75.*
- *Windows installation procedure on page 3-77.*

## 3.1 Linux installation procedure

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

It contains the following sections:

- [Linux software requirements on page 3-75.](#)
- [Linux installation on page 3-75.](#)
- [Linux environment configuration scripts on page 3-75.](#)

### 3.1.1 Linux software requirements

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

<b>Host system</b>	Red Hat Enterprise Linux version 4 or 5. Adobe Acrobat Reader.
<b>License management utilities</b>	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](mailto:license.support@arm.com).

### 3.1.2 Linux installation

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

```
gunzip ModelDebugger_version.tgz
tar -xvf ModelDebugger_version.tar
cd ModelDebugger_version
./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:

<b>bin</b>	Executables.
<b>doc</b>	Documentation.
<b>etc</b>	Model Debugger setup scripts.
<b>lib</b>	Libraries and tool-specific files.

### 3.1.3 Linux environment configuration scripts

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:

```
. installation_directory/etc/setup.sh
```

- For C and related shells use:

```
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.

**Table 3-1 Environment variables**

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

## 3.2 Windows installation procedure

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

It contains the following sections:

- *Windows software requirements on page 3-77.*
- *Windows installation on page 3-77.*

### 3.2.1 Windows software requirements

Model Debugger requires the following software components to run correctly:

<b>Host system</b>	Windows XP with Service Pack 2 or higher, or Microsoft Windows 7 with Service Pack 1 (32-bit and 64-bit).  Adobe Acrobat Reader.
<b>License management utilities</b>	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](mailto:license.support@arm.com).

### 3.2.2 Windows installation

To install Model Debugger:

#### Procedure

1. Open the distribution archive `ModelDebugger_version.zip` and extract the complete contents into a temporary directory. *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.

The installer creates subdirectories in the specified installation directory.

<b>bin</b>	Executables.
<b>doc</b>	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, either copy the value of the `%MAXVIEW_HOME%` environment 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 4

# Shortcuts

This chapter describes shortcuts available in Model Debugger.

It contains the following sections:

- *Keyboard shortcuts on page 4-79.*

## 4.1 Keyboard shortcuts

A table lists the keyboard shortcuts.

**Table 4-1 Keyboard shortcuts**

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

**Table 4-1 Keyboard shortcuts (continued)**

<b>Modifier</b>	<b>Key</b>	<b>Function</b>
Ctrl	P	Open the Model Parameter dialog.
Ctrl+Shift	P	Pause/continue source step.
Ctrl	Q	Close Model Debugger.
Ctrl	R	Reset target only.
Ctrl+Shift	R	Reset target and reload application.
Ctrl	S	Save the current session.
Ctrl	T	Select target.
Ctrl	U	User preferences dialog.
Ctrl	W	Close model.