Model Shell for Fast Models

Version 11.8

Reference Manual

arm

100969_1180_00_en

Model Shell for Fast Models

Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
A	31 May 2014	Non-Confidential	New document for Fast Models v9.0, from DUI0457K for v8.2.
B	30 November 2014	Non-Confidential	Update for v9.1.
C	28 February 2015	Non-Confidential	Update for v9.2.
D	31 May 2015	Non-Confidential	Update for v9.3.
E	31 August 2015	Non-Confidential	Update for v9.4.
F	30 November 2015	Non-Confidential	Update for v9.5.
G	29 February 2016	Non-Confidential	Update for v9.6.
H	31 May 2016	Non-Confidential	Update for v10.0.
I	31 August 2016	Non-Confidential	Update for v10.1.
J	11 November 2016	Non-Confidential	Update for v10.2.
K	17 February 2017	Non-Confidential	Update for v10.3.
1100-00	31 May 2017	Non-Confidential	Update for v11.0. Document numbering scheme has changed.
1101-00	31 August 2017	Non-Confidential	Update for v11.1.
1102-00	17 November 2017	Non-Confidential	Update for v11.2.
1103-00	23 February 2018	Non-Confidential	Update for v11.3.
1104-00	22 June 2018	Non-Confidential	Update for v11.4.
11.4.2-00	17 August 2018	Non-Confidential	Update for v11.4.2.
1105-00	23 November 2018	Non-Confidential	Update for v11.5.
1106-00	26 February 2019	Non-Confidential	Update for v11.6.
1107-00	17 May 2019	Non-Confidential	Update for v11.7.
1108-00	05 September 2019	Non-Confidential	Update for v11.8.

undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

www.arm.com

Contents
Model Shell for Fast Models Reference Manual

Preface

This preface introduces the *Model Shell for Fast Models Reference Manual*.

It contains the following:

About this book

This document describes how to use Model Shell to configure and run Component Architecture Debug Interface (CADI)-compliant models. It provides a command-line reference for the tool and describes how to control its behavior by using configuration files. Arm deprecates Model Shell.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction to Model Shell

This chapter describes the main features of Model Shell, a command line tool for configuring and running a CADI-compliant model.

Chapter 2 Model Shell Commands

This chapter describes how to use Model Shell.

Glossary

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

See the *Arm® Glossary* for more information.

Typographic conventions

italic

Introduces special terminology, denotes cross-references, and citations.

bold

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

`monospace`

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

`monospace`

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

`monospace italic`

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

`monospace bold`

Denotes language keywords when used outside example code.

`<and>`

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

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

SMALL CAPITALS

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

Feedback

Feedback on this product

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

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

Feedback on content

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

- The title *Model Shell for Fast Models Reference Manual*.
- The number 100969_1180_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

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

Other information

- *Arm® Developer*.
- *Arm® Information Center*.
- *Arm® Technical Support Knowledge Articles*.
- *Technical Support*.
- *Arm® Glossary*.

Chapter 1
Introduction to Model Shell

This chapter describes the main features of Model Shell, a command line tool for configuring and running a CADI-compliant model.

Note

Arm deprecates Model Shell from Fast Models version 11.2. Use ISIM executables instead.

It contains the following sections:

1.1 About Model Shell

Model Shell is a command line tool for configuring and running Component Architecture Debug Interface (CADI)-compliant models.

Model Shell launches CADI-compliant models and provides:

- Semihosting stdio.
- CADI logging.
- A launch platform for debuggers, profilers, and operating environments.

Model Shell can start a CADI server to enable other debuggers to connect to the model in the following ways:

- The simulation is initialized, but not run. An external debugger must control the simulation (default).
- The simulation is initialized and run immediately. An external debugger can connect to the simulation after it starts.

Model Shell provides semihosting input and output only for standard streams:

- When a CADI server is started, semihosting output goes to the Model Shell console and to all debuggers.
- If a debugger is attached, it performs semihosting input. If not, Model Shell provides the input.

1.2 ISIM targets

An Integrated SIMulator (ISIM) target is an executable binary that can run standalone, without the need for Model Shell or Model Debugger. ISIMs are generated by simgen by statically linking a model with the SystemC framework.

All Model Shell command-line options, except `--model`, can also be used with an ISIM target. Because the model is integrated into the target, there is no requirement to specify the model on the command line.

Related information

Fast Models User Guide

Chapter 2
Model Shell Commands

This chapter describes how to use Model Shell.

It contains the following sections:

2.1 Model Shell command-line syntax

This section describes the correct arrangement for Model Shell commands, for tailoring the behavior of models.

Syntax

To start Model Shell from the command line, type `model_shell` with any options.

```
model_shell -m model [options]
```

model

File name, including `.so` or `.dll` extension, for the model.

Note

Build a `.so` or `.dll` library from the `.sgproj` file for the model, using System Canvas.

options

List of command-line options.

Related references

2.2 Model Shell command-line options on page 2-13

2.3 Configuration file syntax for specifying model parameters on page 2-15

Related information

Fast Models User Guide

2.2 Model Shell command-line options

Use these options to control Model Shell behavior from the command line.

Table 2-1 Model Shell command line options

Long	Short	Description

`--application` *file1 file2*	`-a`	Load application list. Use `-a [`*INST*`=]`*file* to load an application into a system instance:
		`model_shell $MODEL -a multiprocessor.processor0=app1.axf -a multiprocessor.processor1=app2.axf`
		Use * to load the same application image into all the cores in a cluster:
		`model_shell $MODEL -a "multiprocessor.*=app.axf"`
		Unless you specify a core, Model Shell loads the image into the first cluster or instance that can run software.

`--break` *address*	`-b`	Set program breakpoint at the address. Use `-b [`*INST*`=][`*threadid*`:]`*address* to set a breakpoint for the system instance. Multiple `--break`s set multiple breakpoints.
`--cadi-log`	`-L`	Log all CADI calls to an XML log file.
`--cadi-server`	`-S`	Start a CADI server. This enables attaching a debugger to debug targets in the simulation. To shut down the server, return to the command window that you used to start the model and press **Ctrl+C**. The Model Shell process must be in the foreground before you can shut it down.
`--cadi-trace`	`-t`	Enable diagnostic output from CADI calls and callbacks.
`--config-file` *file*	`-f`	Use model parameters from configuration file *file*.
`--cpulimit` *n*	-	Specify the maximum number of host seconds for the simulation to run, excluding simulation startup and shutdown. Fractions of a second can be specified, but the remaining time is only tested to a resolution of 100ms. If *n* is omitted, the default is unlimited.
`--cyclelimit` *n*	-	Specify the maximum number of cycles to run. If *n* is omitted, the default is unlimited.
`--data` *file@address*	-	Specify raw data to load at this address. The full format is: `--data [`*INST*`=]`*file*`@[`*memspace*`:]`*address*
`--dump` *file@address,size*	-	Dump a section of memory into a file at model shutdown. Multiple `--dump`s are possible. The full format is: `--dump [`*INST*`=]`*file*`@[`*memspace*`:]`*address,size*
`--help`	`-h`	List the Model Shell command-line options, and then exit.
`--keep-console`	`-K`	Keep console window open after completion. Microsoft Windows only.
`--list-instances`	-	Print list of target instances to standard output.
`--list-memory`	-	Print memory information for the model to standard output.

Table 2-1 Model Shell command line options (continued)

Long	Short	Description
`--list-params`	`-l`	Print list of target instances and their parameters to standard output. Use this to help identify the correct syntax for configuration files, and to find out what the target supplies.
`--list-regs`	`-`	Print model register information to standard output.
`--model` *file*	`-m`	Model to load. Optional.
`--output` *file*	`-o`	Redirect output from the `--list-instances`, `--list-memory`, `--list-params`, and `--list-regs` commands to a file. The contents of the file are formatted correctly for use as input by the `--config-file` option.
`--parameter` [*instance.*]*parameter=value*	`-C`	Set a parameter to this value. For hierarchical systems, specify the complete name of the parameter.
`--plugin` *file*	`-`	Specify plugins. These plugins or those in environment variable `FM_PLUGINS` are loaded.
`--prefix`	`-P`	Prefix semihosting output with the name of the target instance.
`--print-port-number`	`-`	Prints the port number on which the CADI server is listening.
`--quiet`	`-q`	Suppress Model Shell output.
`--run`	`-R`	Run simulation on load, with a CADI server: `-S --run`. The default is to start the simulation in a stopped state.
`--start` *address*	`-`	Initialize the PC to this application start address, overriding the `.axf` start. The full format is: `--start [`*INST*`=]`*address*
`--stat`	`-`	Print statistics at the end of the simulation.
`--timelimit` *n*	`-T`	Specify the maximum number of wall clock seconds for the simulator to run. If *n* is omitted, the default is unlimited. If *n* is specified as 0, Model Shell initializes the system, loads all applications and data, sets breakpoints and PC, and exits immediately without running the model. Use this option to convert applications to raw binary. For example: `model_shell --timelimit 0 -m mymodel.dll -a app.axf --dump app.raw@0x8000,0x10000`
`--verbose`	`-V`	Enable verbose messages.
`--version`	`-v`	Print the Model Shell version number, then exit.

Related references

2.5 Model Shell shutdown on page 2-17

2.3 Configuration file syntax for specifying model parameters

Text files can configure models for Model Shell from the command line, thus setting many parameters at once.

Syntax

```
model_shell --config-file my_configuration_file.txt …
```

Each line of the configuration file must have the same *instance.parameter=value* syntax as used for command-line assignments.

Include comment lines and blank lines in configuration files with a # character before the comment or blank text.

To generate a configuration file, use the `--list-instances` and `--list-params` options on the command line. The command line can also include parameter assignments.

Example

```
model_shell --list-params --list-instances -C top-mm=0x3 -o file.config -m model.so
```

might generate:

```
# Instances:
# Instance id: instance name (SW: y/n, component, type, version) : description
# instance.parameter=value    #(type, mode)         default = 'value' : description              : [min..max]
#-------------------------------------------------------------------------------
# Instance 0:                (SW: no , NoCore, , 1.0) : Regression test system without PVLIB usage.
   top-p=0x2                  # (int   , init-time) default = '0x2'   : test display name
   top-str="empty"            # (string, init-time) default = 'empty' : test string param
   top-mm=0x3                 # (int   , init-time) default = '0x6'   : test min(2) max(6) param : [0x2..0x6]
# Instance 1: a1             (SW: no , A, , 1.0) :
   a1.p1=0x2                  # (int   , init-time) default = '0x2'   : A parameter p1
   a1.p2=0                    # (bool  , run-time ) default = '0'     : A parameter p2
# Instance 2: a1.b           (SW: no , B, , 1.0) :
   a1.b.p1=0x2                # (int   , init-time) default = '0x2'   : B parameter p1
   a1.b.p2=""                 # (string, run-time ) default = ''      : B parameter p2
# Instance 3: a2             (SW: no , A, , 1.0) :
   a2.p1=0x2                  # (int   , init-time) default = '0x2'   : A parameter p1
   a2.p2=0                    # (bool  , run-time ) default = '0'     : A parameter p2
# Instance 4: a2.b           (SW: no , B, , 1.0) :
   a2.b.p1=0x2                # (int   , init-time) default = '0x2'   : B parameter p1
   a2.b.p2="test"             # (string, run-time ) default = ''      : B parameter p2
#-------------------------------------------------------------------------------
```

This is another way of specifying run-time parameters:

```
# 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
```

2.4 SMP support

Model Shell provides Symmetric MultiProcessing support. It can be simple or standard.

Simple

This is only suitable for model systems that have one SMP cluster. The same application is loaded in all cores.

```
model_shell -m smp_model.so -a app.axf
```

Standard

This is suitable for all cases and uses the `-a` option to list the applications for each core.

Use the full instance name of each core.

```
model_shell -m smp_model.so -a multiprocessor.processor0=app1.axf -a
multiprocessor.processor1=app2.axf
```

In addition to loading individual applications for each core, the `-a` option also enables loading the same application in all cores.

Replace the index of the core with `*`.

```
model_shell -m smp_model.so -a multiprocessor.processor*=app.axf
```

```
model_shell -m smp_model.so -a "multiprocessor.*"=app.axf
```

Note

On Unix, the `*` character requires escape quotes.

2.5 Model Shell shutdown

This section describes the actions that stop Model Shell manually, and the options that stop it automatically.

This section contains the following subsections:

2.5.1 Manual Model Shell shutdown

User actions that stop Model Shell.

Press `Ctrl+C`.[a]
: The program starts shutting down the simulator and exits after shutdown is complete. On a second press, Model Shell terminates immediately.

Press `Ctrl+Break`.[b]
: Model Shell terminates immediately.

Close an LCD window.
: The simulation stops.

2.5.2 Automatic Model Shell shutdown

Command-line options that define when to stop Model Shell.

None[c]
: Simulation end.

`--break`[c]
: Breakpoint.

`--cyclelimit`[cd]
: Cycles > cycle limit.

`--timelimit`
: Time > running time limit.

`--cpulimit`[e]
: Time > process time limit.

Note

The first fulfilled condition stops Model Shell.

---

a Some models can assign their own `Ctrl+C` handlers that override Model Shell behavior.
b Windows only.
c `--cadi-server` overrides this.
d
- Might reduce execution speed.
- Ignores breakpoints.

e
- Tested to a granularity of 0.1s to avoid performance loss.
- Elapsed processor time includes user time and kernel time.

2.6 License checking messages from Model Shell and ISIM systems

The license checking messages appear in the `stderr` and `stdout` outputs, and are useful for the detection and diagnosis of licensing issues.

The `model_shell` and `isim_system` executables return a status value when they exit:

`0`
: no error (for example, clean simulator shutdown).

`1`
: error (for example, license checking or file not found).

For exit status 1, parse the `stderr` output. A message might, for example, appear in the GUI with other `WARNING`, `ERROR` or `Fatal Error` messages. See the lines that follow for more information from the license checking module. When a license is about to expire, Model Shell prints a warning message to `stdout`, but the simulation still starts correctly.

Example

```
ERROR: License check failed!
Either the license file or the license server could not be found.
Please set the environment variable 'ARMLMD_LICENSE_FILE'
to your license file location or refer to the ARM_FLEXnet_Guide
for instructions on where to obtain a license file, where to install
the license file, and how to setup a license server.
Error Code : -1
```