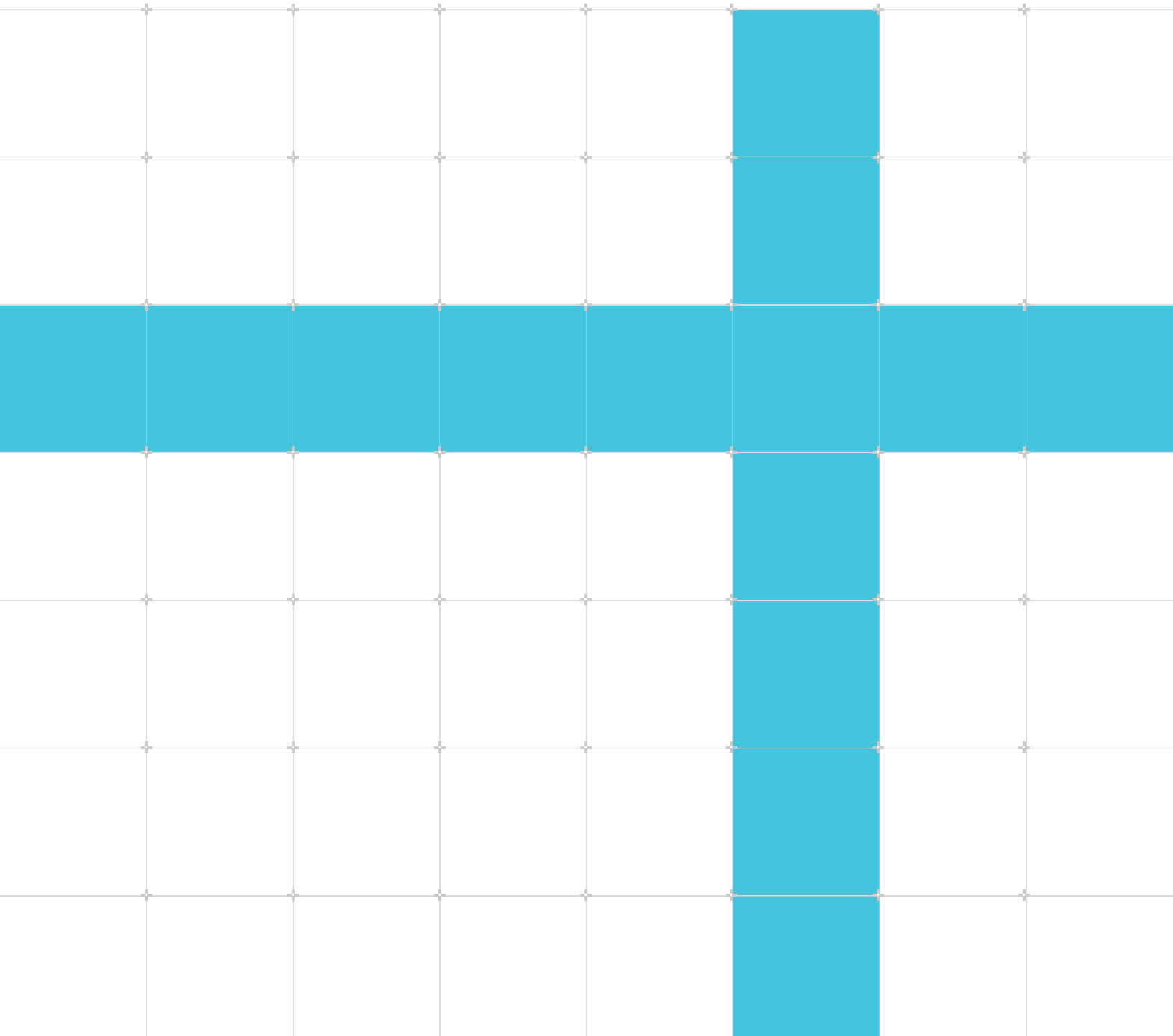
arm

Model Shell for Fast Models

Version 11.24

Reference Guide

Issue 00
100969_1124_00_en



Model Shell for Fast Models

Reference Guide

Release Information

Document history

Issue	Date	Confidentiality	Change
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.
1104-01	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	5 September 2019	Non-Confidential	Update for v11.8.
1109-00	28 November 2019	Non-Confidential	Update for v11.9.
1110-00	12 March 2020	Non-Confidential	Update for v11.10.
1111-00	9 June 2020	Non-Confidential	Update for v11.11.

Issue	Date	Confidentiality	Change
1112-00	22 September 2020	Non-Confidential	Update for v11.12.
1113-00	9 December 2020	Non-Confidential	Update for v11.13.
1114-00	17 March 2021	Non-Confidential	Update for v11.14.
1115-00	29 June 2021	Non-Confidential	Update for v11.15.
1116-00	6 October 2021	Non-Confidential	Update for v11.16.
1117-00	16 February 2022	Non-Confidential	Update for v11.17.
1118-00	15 June 2022	Non-Confidential	Update for v11.18.
1119-00	14 September 2022	Non-Confidential	Update for v11.19.
1120-00	7 December 2022	Non-Confidential	Update for v11.20.
1121-00	22 March 2023	Non-Confidential	Update for v11.21.
1122-00	14 June 2023	Non-Confidential	Update for v11.22.
1123-00	13 September 2023	Non-Confidential	Update for v11.23.
1124-00	6 December 2023	Non-Confidential	Update for v11.24.

Proprietary Notice

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, 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.

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|version 21.0)

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com.

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction

This document describes how to use Model Shell to configure and run Component Architecture Debug Interface (CADI)-compliant models.

1.1 Conventions

The following subsections describe conventions used in Arm documents.

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: developer.arm.com/glossary.

Convention	Use
italic	Citations.
bold	Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<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	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

Recommendations. Not following these recommendations might lead to system failure or damage.

Requirements for the system. Not following these requirements might result in system failure or damage.

Requirements for the system. Not following these requirements will result in system failure or damage.

An important piece of information that needs your attention.

A useful tip that might make it easier, better or faster to perform a task.

A reminder of something important that relates to the information you are reading.

1.2 Other information

See the Arm® website for other relevant information.

- Arm® Developer.
- Arm® Documentation.
- Technical Support.
- Arm® Glossary.

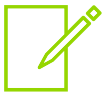
1.3 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Fast Models User Guide	100965	Non-Confidential



Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com.

2. 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 does not run. An external debugger must control the simulation. This is the default behavior.
- The simulation is initialized and runs 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.

Model Shell is included in the Fast Models package and in the Fast Models FVP Standard Library package. For installation information for the Fast Models package, see Installation in the *Fast Models User Guide*.

2.1 ISIM targets

An Integrated SIMulator (ISIM) target is a standalone executable that runs without the need for Model Shell or Model Debugger. SimGen generates ISIMs by statically linking a specific model library with the SystemC framework.

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

Related information

Fast Models User Guide

2.2 Model Shell command-line syntax

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

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

`model`

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

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

`options`

List of command-line options.

Related information

Model Shell command-line options on page 11
Configuration file syntax for specifying model parameters on page 13
Fast Models User Guide

2.3 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 cluster0.cpu0=app1.axf -a cluster0.cpu1=app2.axf`
		
		Use `*` to load the same application image into all the cores in a cluster:
		
		`model_shell $MODEL -a "cluster0.*=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 `--breaks` 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 a configuration file.

Long	Short	Description
`--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 `--dumps` 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.
`--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 [inst.]param=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.

Long	Short	Description
`--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 information

Automatic Model Shell shutdown on page 15

2.4 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.

```
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
```

2.5 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.

2.6 Manual Model Shell shutdown

User actions that stop Model Shell.

Press `Ctrl+C`.

The program starts shutting down the simulator and exits after shutdown is complete. On a second press, Model Shell terminates immediately. Some models can assign their own **`Ctrl+C`** handlers that override Model Shell behavior.

Press `Ctrl+Break`.

Model Shell terminates immediately. Windows only.

Close an LCD window.

The simulation stops.

2.7 Automatic Model Shell shutdown

Command-line options that control when to stop Model Shell. The first fulfilled condition stops Model Shell. If none are specified, Model Shell shuts down when the simulation ends.

`--break`

Breakpoint. `--cadi-server` overrides this option.

`--cyclelimit`

Cycles > cycle limit. `--cadi-server` overrides this option. This option might reduce execution speed. It ignores breakpoints.

`--timelimit`

Time > running time limit.

`--cpulimit`

Time > process time limit. Tested to a granularity of 0.1s to avoid performance loss. Elapsed processor time includes user time and kernel time.

2.8 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.