



Foundation Platform

Version 11.22

User Guide

Non-Confidential

Copyright © 2017–2023 Arm Limited (or its affiliates).
All rights reserved.

Issue 00

100961_1122_00_en



Foundation Platform User Guide

Copyright © 2017–2023 Arm Limited (or its affiliates). All rights reserved.

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	27 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.
1108-01	3 October 2019	Non-Confidential	Update for v11.8.1.
1109-00	28 November 2019	Non-Confidential	Update for v11.9.
1110-00	12 March 2020	Non-Confidential	Update for v11.10.

Issue	Date	Confidentiality	Change
1111-00	9 June 2020	Non-Confidential	Update for v11.11.
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.

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

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.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2017–2023 Arm Limited (or its affiliates). All rights reserved.

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.....	7
1.1 Conventions.....	7
1.2 Other information.....	8
1.3 Useful resources.....	8
2. Foundation Platform introduction.....	10
2.1 Host requirements.....	10
2.2 Platform overview.....	10
2.2.1 Features of the Foundation Platform.....	11
2.2.2 Processor models in the Foundation Platform.....	12
2.2.3 Network support in the Foundation Platform.....	12
2.2.4 Foundation Platform block diagram.....	13
2.2.5 Limitations of the Foundation Platform.....	13
3. Getting Started.....	15
3.1 Contents of the installation.....	15
3.2 Running the example program.....	16
3.3 Troubleshooting the example program.....	16
3.4 Using Linux.....	17
4. Programming Reference.....	18
4.1 Command-line options.....	18
4.2 SVE-related command-line options.....	20
4.3 Foundation Platform memory map.....	21
4.4 Clock and timer.....	23
4.5 Interrupt maps.....	24
4.6 System register block.....	25
4.7 CLCD window.....	27
4.8 UARTs.....	29
4.9 Multicore configuration.....	30
4.10 Semihosting.....	31

1. Introduction

This document describes the Foundation Platform for Arm A-class architectures. The Foundation Platform is a free of charge Fixed Virtual Platform that aids Linux application development.

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
<i>italic</i>	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 <u>underline</u>	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: <pre>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></pre>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE .



Caution

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



Warning

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
Arm® Development Studio User Guide	101470	Non-Confidential
Open Source Software and Platforms wiki	-	Non-Confidential
Fast Models on Arm® Developer	-	Non-Confidential
Fast Models Reference Guide	100964	Non-Confidential
Iris Python Debug Scripting User Guide	101421	Non-Confidential

Arm® architecture and specifications	Document ID	Confidentiality
A-Profile Architecture Specifications	-	Non-Confidential
Semihosting for AArch32 and AArch64 specification	-	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. Foundation Platform introduction

The Foundation Platform is an enabling platform for the Arm®v8-A and Armv9-A architectures.

It is a simple platform model capable of running bare-metal semi-hosted applications and booting a full operating system, with processor cluster, RAM, and some basic I/O devices such as Universal Asynchronous Receiver/Transmitters (UARTs), block storage, and network support. It also contains a simple web interface to indicate the status of the platform. It is supplied as a platform with configuration of the simulation from the command line and control using peripherals in the platform.

The Foundation Platform is a Programmers View (PV) model, which sacrifices timing accuracy to achieve fast simulation execution speeds. You can use it for confirming software functionality, but you must not rely on the accuracy of cycle counts, low-level component interactions, or other hardware-specific behavior.

It uses Arm® Fast Models technology and forms part of a comprehensive suite of modeling solutions for Arm® processors. These modeling solutions are available in the portfolio of models that are delivered through the Arm® Fast Models product. For more information, see [Fast Models](#) on Arm Developer.

2.1 Host requirements

This section describes requirements for the host machine on which the Foundation Platform runs.

Operating Systems

Foundation Platform supports x86-64 and Arm® AArch64 host platforms running Red Hat Enterprise Linux 7 or 8 (for 64-bit architectures), or Ubuntu 18.04 or 20.04 Long Term Support (LTS).

UART output

For the *Universal Asynchronous Receiver/Transmitter* (UART) output to be visible, both `xterm` and `telnet` must be installed on the host, and be specified in your PATH.

Visualization

The Foundation Platform must run in an environment where visualization is possible. For example, running it on a Linux terminal without an X server is not supported.

2.2 Platform overview

This section describes the features and limitations of the Foundation Platform, and the types of network support that are provided.

2.2.1 Features of the Foundation Platform

The Foundation Platform has the following features:

- A cluster model containing 1-4 AEMvA cores. AEMvA is an Architecture Envelope Model (AEM) that implements all architectural features in both Arm®v8-A and Armv9-A.
- Up to 8GB of RAM. To simulate a system with 4GB of RAM, you require a host with at least 8GB of RAM. To simulate a system with 8GB of RAM, you require a host with at least 12GB of RAM.
- Four PL011 UARTs connected to xterms.
- Platform peripherals including a real-time clock, watchdog timer, real-time timer, and power controller.
- Secure peripherals including a trusted watchdog, random number generator, non-volatile counters, and root-key storage.
- A network device model that is connected to host network resources.
- A block storage device that is implemented as a file on the host.
- A small system register block with LEDs and switches visible using a web server.
- Host filesystem access that is implemented as a Plan 9 filesystem.
- A CLCD that allows GUI visualization.
- Debug capabilities through the use of a CADI server.
- TarmacTrace support is built into the model. To enable it, use the `--trace=<file>` command-line option.

Caches are modeled as stateless and there are no write buffers. This gives the effect of perfect memory coherence on the data side. The instruction side has a variable size prefetch buffer so requires correct barriers to be used in target code to operate correctly.

The platform runs as fast as possible unless all the cores in the cluster are *Wait for Interrupt* (WFI) or *Wait for Exception* (WFE). In the case of WFE, the platform idles until an interrupt or external event occurs.

The Foundation Platform has been revised to support the Arm® *Trusted Base System Architecture* (TBSA) and *Server Base System Architecture* (SBSA). Several peripheral devices have been added, with corresponding changes to the memory map. It has also been updated to align more closely with peripherals present in the Versatile™ Express baseboard and in Arm® Fast Models.

Software that is written to target the previous versions of the platform work unmodified on the platform by using the `--no-gicv3` configuration option. Only software that uses the early blocks of RAM is likely to require some adjustments.

Related information

[Foundation Platform memory map](#) on page 21

[Command-line options](#) on page 18

[A-Profile Architecture Specifications](#)

2.2.2 Processor models in the Foundation Platform

The processor models in this platform are not based on any existing processor design, but conform to the Arm®v8-A and Armv9-A architectural specifications.

They implement:

- Arm®v8-A versions 8.0-8.7.
- Armv9-A, versions 9.0-9.2.
- AArch64 at all exception levels.
- AArch32 support at EL0 and EL1.
- Little and big-endian support at all exception levels.
- Generic timers.
- Self-hosted debug.
- GICv2 and GICv3 memory-mapped processor interfaces and distributor.
- Scalable Vector Extension (SVE) and SVE2.

2.2.3 Network support in the Foundation Platform

The platform provides the following types of network support:

NAT, IPv4 based

NAT, IPv4-based networking provides limited IP connectivity by using user-level IP services. This requires no extra privileges to set up or use, but has inherent limitations. System-level services, or services conflicting with those services on the host, can be provided using port remapping.

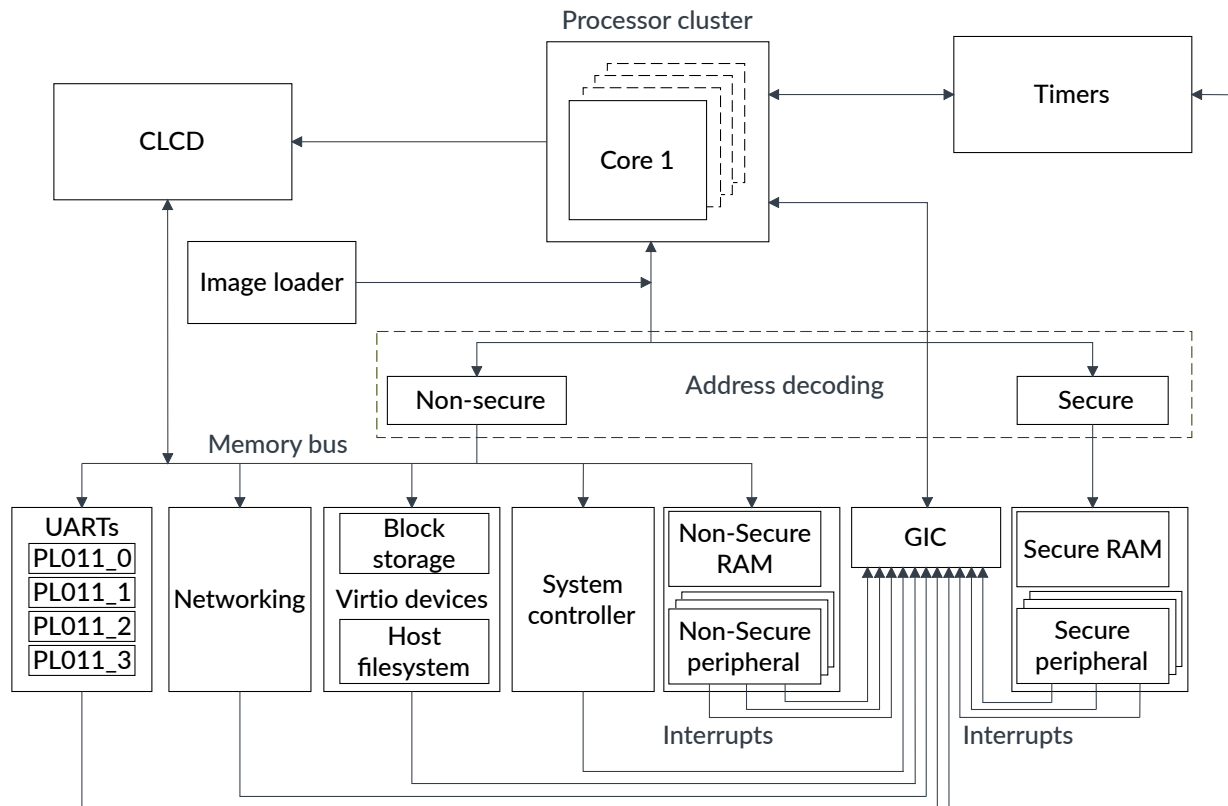
Bridged

Bridged networking requires the setup of an ethernet bridge device to bridge between the ethernet port on the host and the network interface that the platform provides. This usually requires administrator privileges. See the documentation in the Linux bridge-utils package for more information.

2.2.4 Foundation Platform block diagram

This figure shows the block diagram for the Foundation Platform.

Figure 2-1: Foundation Platform block diagram



The behavior of the address decoding block depends on whether the `--secure-memory` command-line option is used.

2.2.5 Limitations of the Foundation Platform

The following restrictions apply to the Foundation Platform:

- Write buffers are not modeled.
- Interrupts are not taken at every instruction boundary.
- Caches are modeled as stateless.
- Plug-ins are not supported, although TarmacTrace support is built into the Foundation Platform.

- There is no support for Thumb®2EE.
- There is no support for the Arm®v8 cryptography extensions.
- Arm does not offer direct customer support for the Foundation Platform. For technical support use the Arm® Connected Community, <http://community.arm.com>.

Related information

[Command-line options](#) on page 18

3. Getting Started

This chapter describes testing the Foundation Platform installation.

3.1 Contents of the installation

The Foundation Platform is supplied as a prebuilt platform binary, along with some other libraries and tools.

The following list describes the contents of each top-level directory after installation:

iris.debug

A Python module that allows you to interact with the Foundation Platform, including connecting to and configuring it, performing execution control, and accessing registers and memory. It is installed under the `iris` directory. For more information about `iris.debug`, see [Iris Python Debug Scripting User Guide](#).

bin

Contains the `arm1m` tool for managing User-Based Licensing. For details, see [User-Based Licensing User Guide](#).

doc

Contains a readme file and a PDF version of this document. The readme provides an overview of the platform and release notes for this and previous platform versions.

examples

Contains source code and executable for an example Hello World program. It also includes the Device Tree Source (DTS) file for the Foundation Platform.

fmtplib

Contains the GCC runtime libraries to allow you to run the Foundation Platform on older Linux distributions.

license_terms

Contains the license information for third-party software and the end-user license agreement.

models

Contains the Foundation Platform executable and shared objects that must be present in the same directory as the executable when running it.

scripts

Contains a script to source if you need to add the location of the GCC runtime libraries installed in `fmtplib/` to `LD_LIBRARY_PATH`.

Related information

[Running the example program](#) on page 16

[Device Tree](#)

3.2 Running the example program

You can use the example program that is supplied to confirm that the Foundation Platform is working correctly.

Procedure

1. Run the example from the command line. For example, from the directory containing the Foundation Platform executable, run: **`./Foundation_Platform --image ../../examples/hello.axf`**
2. Optionally add `--quiet` to suppress all output except for the output from the program. The program should produce output similar to this:

```
terminal_0: Listening for serial connection on port 5000
terminal_1: Listening for serial connection on port 5000
terminal_3: Listening for serial connection on port 5000
terminal_2: Listening for serial connection on port 5000
Hello, 64-bit world!
```

This demonstrates that the platform initialized correctly as it loaded and executed the example program. It also demonstrates that the semihosting calls to print output and stop the platform worked properly.

3.3 Troubleshooting the example program

You might encounter common error messages when running the example program.

- If you attempt to run the example program on a 32-bit Linux host, it gives an error similar to the following:

```
./Foundation_Platform: /lib64/ld-linux-x86-64.so.2: bad ELF interpreter: No such file or directory
```

- If `libstdc++` is not installed on your system, you get the following error on startup:

```
./Foundation_Platform: error while loading shared libraries: libstdc++.so.6: cannot open shared object file
```

- If your system `glibc` or `libstdc++` is too old, you might see a message like the following:

```
./Foundation_Platform: /usr/lib/x86_64-linux-gnu/libstdc++.so.6: version `GLIBCXX_3.4.26' not found (required by ./Foundation_Platform)
```

To resolve issues with missing or out of date GCC runtime libraries, source the `runtime.sh` script located in the `scripts` directory.

3.4 Using Linux

Arm provides validated Linux and Android deliverables for the Foundation Platform.

These are available on the Arm Community website at [Arm Development Platforms](#). To get started with Linux on the Foundation Platform and other FVPs, see [FVPs](#) on Arm Community.

4. Programming Reference

This chapter contains reference information for the Foundation Platform.

4.1 Command-line options

Command-line options provide all platform configuration. To see a summary of the available commands, run the platform with `--help`.

The syntax to use on the command line is:

```
./Foundation_Platform [OPTIONS...]
```

Table 4-1: Command-line options

Option	Description
<code>--arm-v8.n</code>	Enable the Arm®v8. <i>n</i> version of the architecture, where $0 \leq n \leq 7$.
<code>--arm-v9.n</code>	Enable the Armv9. <i>n</i> version of the architecture, where $0 \leq n \leq 2$. The default is <code>--arm-v9.2</code> , which also enables Arm®v8.7. Note: Enabling <code>--arm-v9.1</code> also enables Arm®v8.6, and enabling <code>--arm-v9.0</code> also enables Arm®v8.5.
<code>--bigendian</code>	Start processors in big-endian mode. The default is little-endian.
<code>--block-device=file</code>	Image file to use as persistent block storage.
<code>--cadi-server</code>	Start the CADI server. This option allows a CADI-enabled debugger to connect to targets in the simulation.
<code>--cores=N</code>	Specify the number of processors, where <i>N</i> is in the range 1-4. The default is 1.
<code>--(ns) data=file@address</code>	Raw file to load at an address in non-secure or secure memory. Use <code>--nsdata</code> to load data into non-secure memory when the <code>--secure-memory</code> option is enabled.
<code>--(no-) gicv3</code>	Enable GICv3 or the legacy, compatible GICv2. The default is <code>--gicv3</code> .
<code>--help</code>	Display the command-line options and quit.
<code>--image=file</code>	ELF image to load.
<code>--min-sync-latency=N</code>	Number of ticks to simulate before synchronizing. Events that occur at a higher frequency than this value are missed. The default is 100.
<code>--network=(none nat bridged)</code>	Configure the network access mode. The default is none.
<code>--network-bridge=dev</code>	Bridged network device name. The default is ARM0.
<code>--network-mac-address</code>	MAC address to use for networking. The default is 00:02:f7:ef:f6:74.
<code>--network-nat-ports=M</code>	Optional comma-separated list of NAT port mappings in the form: <i>host_port=model_port</i> , for example, 8022=22.
<code>--network-nat-subnet=S</code>	Subnet used for NAT networking. The default is 172.20.51.0/24.
<code>--p9_root_dir=dir</code>	Host folder to be shared between the host and the guest.

Option	Description
<code>--print-port-number</code>	Print the port number that the CADI server is listening to. This option can help with identifying the debug server to connect to when multiple servers are enabled.
<code>--quiet</code>	Suppress any non-simulated output on <code>stdout</code> or <code>stderr</code> .
<code>--quantum=N</code>	Number of ticks to simulate for each quantum. The default is 10000.
<code>--(no-)rate-limit</code>	Enable or disable rate limiting. Enabling rate limiting restricts the simulation speed so that simulation time more closely matches real time rather than running as fast as possible. The default is disabled.
<code>--read-only</code>	Mount block device image in read-only mode.
<code>--(no-)secure-memory</code>	Enable or disable separate secure and non-secure address spaces. The default is disabled.
<code>--(no-)semihost</code>	Enable or disable semihosting support. The default is enabled.
<code>--semihost-cmd=cmd</code>	A string that is used as the semihosting command line.
<code>--semihosting-heap_base=address</code>	Virtual address of the heap base. The default is 0.
<code>--semihosting-heap_limit=address</code>	Virtual address of the top of the heap. The default is 0xFF000000.
<code>--semihosting-stack_base=address</code>	Virtual address of the base of the descending stack. The default is 0xFFFF0000.
<code>--semihosting-stack_limit=address</code>	Virtual address of the stack limit. The default is 0xFF000000.
<code>--(no-)sve</code>	Enable or disable <i>Scalable Vector Extension</i> (SVE) and SVE2. The default is enabled. This option requires Arm®v8.2 or a later architecture to be enabled. If SVE is enabled, the options described in 4.2 SVE-related command-line options on page 19 can be used.
<code>--switches=val</code>	Initial setting of switches in the system register block. The default is 0.
<code>--trace=file</code>	Enable the TarmacTrace extension. Unlike the TarmacTrace plug-in, the extension is not configurable. Its behavior is the same as the default behavior of the TarmacTrace plug-in, as described in TarmacTrace parameters , except that the extension requires you to specify a trace output file.
<code>--uartN-outfile=file</code>	Redirect output from UART N to a file, where N is in the range 0-3. Specify a filename of <code>-</code> to redirect output to <code>stdout</code> . If no filename is specified, the option is ignored.
<code>--uart-start-port=P</code>	Attempt to listen on a free TCP port in the range P to $P+100$ for each UART. The default is 5000.
<code>--use-real-time</code>	Sets the generic timer registers to report a view of real time as it is seen on the host platform, instead of simulated time.
<code>--version</code>	Display the version and build numbers and quit.

You can specify more than one `--image`, `--data`, or `--nsdata` option. The images and data are loaded in the order that they appear on the command line. The simulation starts from the entry point of the final ELF file specified.

Related information

[Multicore configuration](#) on page 30

[Semihosting](#) on page 30

4.2 SVE-related command-line options

These additional command-line options only take effect if the Scalable Vector Extension (SVE) is enabled, which is controlled by the `--(no-)sve` option. By default SVE is enabled.



To enable or disable support for particular features, use the `--has-*` parameters.

Table 4-2: Additional SVE-related options

Option	Type	Description
<code>--clear-constrained-lanes</code>	int	When a constrained vector length increases, previously inaccessible bits are set to zero according to the value of this parameter. Possible values are: 0x0 Never. This is the default. 0x1 Always 0x2 If the register was written to while the vector length was constrained.
<code>--combine-movprfx-and-destructive</code>	bool	Attempt to combine the execution of <code>MOVPRFX</code> and the destructively encoded instruction that follows it. The default is false.
<code>--disable-speculative-accesses</code>	bool	All speculative memory accesses behave as though faulting without accessing memory. The default is false.
<code>--enable-at-reset</code>	bool	Start with system registers set up for Scalable Vector Extension use. The default is false.
<code>--ffr-16b-pattern-UNKNOWN</code>	int	A specific 16-bit UNKNOWN value that is used by parameter <code>force_UNKNOWN_to_ffr</code> . The default is 0.
<code>--force-UNKNOWN-to-ffr</code>	int	Governs the behavior if <code>WRFFR</code> writes a non-monotonic value to <code>FFR</code> . Possible values are: 0x0 Write non-canonical value to <code>FFR</code> . This is the default. 0x1 Overwrite <code>FFR</code> with a specific 16-bit UNKNOWN value. See <code>ffr_16b_pattern_UNKNOWN</code> . 0x2 Clear all bits above first zero.
<code>--fp-exception-report-lowest</code>	bool	If true, for multiple trapped FP exceptions, report the lowest lane in <code>VECITR</code> . Otherwise, report the highest. The default is false.
<code>--fp-exception-set-tfv</code>	bool	Set <code>ESR_ELx.TFV</code> during FP exception. Trapped exception flags are valid. The default is true.
<code>--fp-exception-set-vecitr</code>	bool	If true, set <code>ESR_ELx.VECITR</code> during FP exception. Otherwise, set <code>RES0</code> . The default is false.
<code>--has-sve2</code>	bool	Whether SVE2 is implemented. The default is false.
<code>--has-sve2-aes</code>	int	If SVE2 is implemented, whether SVE2 AES instructions are implemented. Possible values are: 0x0 Not implemented. 0x1 SVE2 <code>AESE</code> , <code>AESD</code> , <code>AESMC</code> , and <code>AESIMC</code> are implemented. 0x2 Same as 1, but in addition, SVE2 <code>PMULLB</code> and <code>PMULLT</code> with 64-bit source are implemented. This is the default.
<code>--has-sve2-bit-perm</code>	bool	If SVE2 is implemented, whether BitPerm instructions are implemented. The default is true.
<code>--has-sve2-sha3</code>	bool	If SVE2 is implemented, whether SHA3 instructions are implemented. The default is true.
<code>--has-sve2-sm4</code>	bool	If SVE2 is implemented, whether SM4 instructions are implemented. The default is true.
<code>--has-sve-bf16</code>	bool	Whether SVE BFloat16 instructions are implemented. The default is true.

Option	Type	Description
--has-sve-mm-f32	bool	Whether the SVE FP32 Matrix Multiply instructions are implemented. The default is true.
--has-sve-mm-f64	bool	Whether the SVE FP64 Matrix Multiply instructions are implemented. The default is true.
--has-sve-mm-i8	bool	Whether the SVE Int8 Matrix Multiply instructions are implemented. The default is true.
--movprfx-unpredictable-behavior	int	Defines the behavior of MOVPRFX and the instruction it immediately precedes when the behavior is CONSTRAINED UNPREDICTABLE . Possible values are: 0 UNDEF execution from MOVPRFX. This is the default. 1 MOVPRFX and second half of instruction executes as NOP. 2 NOP MOVPRFX only.
--predicated-sp-align-check-behaviour	int	Governs behavior of SP alignment checking for predicated memory accesses. Possible values are: 0x0 Always perform. This is the default. 0x1 Skip if governing predicate is 0. 0x2 Skip for contiguous accesses if governing predicate is 0. 0x3 Skip for gather-scatter accesses if governing predicate is 0.
--support-npot-vl	bool	Whether vector lengths that are not a power of two are supported. The default is true.
--undef-invalid-combined-movprfx	bool	If a combined MOVPRFX is invalid, raise an UNDEF exception. Otherwise NOP the second half of the register. This parameter is deprecated. The default is true.
--unknown-value	int	Simulated value for a state that has an UNKNOWN value after reset. The default is 0xdeadbeefdeadbeef.
--veclen	int	Size of the vector in units of 64-bit blocks. Allowed range is 0x2-0x20. The default is 8.
--z-reg-on-load-fault-behaviour	int	Governs the behavior of destination Z-registers in case of a load fault. Possible values are: 0x0 Register becomes UNKNOWN . This is the default. 0x1 Register is preserved.

4.3 Foundation Platform memory map

This section describes the memory map for the Foundation Platform.

The following list shows the Secure and Non-secure access permissions that are enabled by using the --(no-)secure-memory parameter.

Table 4-3: Access permissions

	--no-secure-memory	--secure-memory
S	Secure and Non-secure accesses are permitted.	Secure access is permitted, Non-secure access aborts.
S/NS	Secure and Non-secure accesses are permitted.	Secure and Non-secure accesses are permitted.

The following table shows the global memory map for the Foundation Platform. This map is based on the Versatile™ Express RS2 memory map with extensions.



- Unless you use the --quiet command-line option, areas of memory that are highlighted in the table return a warning to the console, together with RAZ/WI access behavior.

- Writes are ignored.

Table 4-4: Foundation Platform memory map

Start address	End address	Peripheral	Size	Security
0x00_0000_0000	0x00_03FF_FFFF	Trusted Boot ROM, secureflash	64MB	S
0x00_0400_0000	0x00_0403_FFFF	Trusted SRAM	256KB	S
0x00_0600_0000	0x00_07FF_FFFF	Trusted DRAM	32MB	S
0x00_0800_0000	0x00_0BFF_FFFF	NOR flash, flash0	64MB	S/NS
0x00_0C00_0000	0x00_0FFF_FFFF	NOR flash, flash1	64MB	S/NS
0x00_1800_0000	0x00_19FF_FFFF	VRAM	32MB ¹	-
0x00_1A00_0000	0x00_1AFF_FFFF	Ethernet, SMSC 91C111	16MB	S/NS
0x00_1C01_0000	0x00_1C01_FFFF	System Registers	64KB	S/NS
0x00_1C02_0000	0x00_1C02_FFFF	System Controller, SP810	64KB	S/NS
0x00_1C04_0000	0x00_1C07_FFFF	Warning + RAZ/WI	-	-
0x00_1C09_0000	0x00_1C09_FFFF	UART0, PL011	64KB	S/NS
0x00_1C0A_0000	0x00_1C0A_FFFF	UART1, PL011	64KB	S/NS
0x00_1C0B_0000	0x00_1C0B_FFFF	UART2, PL011	64KB	S/NS
0x00_1C0C_0000	0x00_1C0C_FFFF	UART3, PL011	64KB	S/NS
0x00_1C0D_0000	0x00_1C0D_FFFF	Warning + RAZ/WI	-	-
0x00_1C0F_0000	0x00_1C0F_FFFF	Watchdog, SP805	64KB	S/NS
0x00_1C10_0000	0x00_1C10_FFFF	Base Platform Power Controller	64KB	S/NS
0x00_1C11_0000	0x00_1C11_FFFF	Dual-Timer 0, SP804	64KB	S/NS
0x00_1C12_0000	0x00_1C12_FFFF	Dual-Timer 1, SP804	64KB	S/NS
0x00_1C13_0000	0x00_1C13_FFFF	Virtio block device	64KB	S/NS
0x00_1C14_0000	0x00_1C14_FFFF	Virtio Plan 9 for v9, Warning + RAZ/W for v2.1	64KB	S/NS
0x00_1C15_0000	0x00_1C15_FFFF	Virtio net device	64KB	S/NS
0x00_1C17_0000	0x00_1C17_FFFF	Realtime Clock, PL031	64KB	S/NS
0x00_1C1A_0000	0x00_1FFF_FFFF	Warning + RAZ/W	-	-
0x00_1F00_0000	0x00_1F00_0FFF	Non-trusted ROM	4KB	S/NS
0x00_2A43_0000	0x00_2A43_FFFF	REFCLK CNTControl, Generic Timer	64KB	S
0x00_2A44_0000	0x00_2A44_FFFF	EL2 Generic Watchdog Control	64KB	S/NS
0x00_2A45_0000	0x00_2A45_FFFF	EL2 Generic Watchdog Refresh	64KB	S/NS
0x00_2A49_0000	0x00_2A49_FFFF	Trusted Watchdog, SP805	64KB	S
0x00_2A4A_0000	0x00_2A4A_FFFF	Warning + RAZ/W	-	-
0x00_2A80_0000	0x00_2A80_FFFF	REFCLK CNTRead, Generic Timer	64KB	S/NS
0x00_2A81_0000	0x00_2A81_FFFF	AP_REFCLK CNTCTL, Generic Timer	64KB	S/NS
0x00_2A82_0000	0x00_2A82_FFFF	AP_REFCLK CNTBase0, Generic Timer	64KB	S
0x00_2A83_0000	0x00_2A83_FFFF	AP_REFCLK CNTBase1, Generic Timer	64KB	S/NS

¹ 8MB of VRAM is replicated 4 times in memory.

Start address	End address	Peripheral	Size	Security
0x00_2C00_0000	0x00_2C00_1FFF	GIC Physical CPU interface, GICC ²	8KB	S/NS
0x00_2C00_1000	0x00_2C00_1FFF	GIC Distributor ³	4KB	-
0x00_2C00_2000	0x00_2C00_2FFF	GIC Processor Interface ³	4KB	-
0x00_2C00_4000	0x00_2C00_4FFF	GIC Processor Hyp Interface ³	4KB	-
0x00_2C00_5000	0x00_2C00_5FFF	GIC Hyp Interface ³	4KB	-
0x00_2C00_6000	0x00_2C00_7FFF	GIC Virtual CPU Interface ³	8KB	-
0x00_2C01_0000	0x00_2C01_0FFF	GIC Virtual Interface Control, GICH	4KB	S/NS
0x00_2C02_F000	0x00_2C03_0FFF	GIC Virtual CPU Interface, GICV	8KB	S/NS
0x00_2C09_0000	0x00_2C09_FFFF	Warning + RAZ/W	-	-
0x00_2E00_0000	0x00_2E00_FFFF	Non-trusted SRAM	64KB	S/NS
0x00_2F00_0000	0x00_2F00_FFFF	GICv3 Distributor GICD ²	64KB	S/NS
0x00_2F10_0000	0x00_2F1F_FFFF	GICv3 Distributor GICR	1MB	S/NS
0x00_7FE6_0000	0x00_7FE6_0FFF	Trusted Random Number Generator	4KB	S
0x00_7FE7_0000	0x00_7FE7_0FFF	Trusted Non-volatile counters	4KB	S
0x00_7FE8_0000	0x00_7FE8_0FFF	Trusted Root-Key Storage	4KB	S
0x00_8000_0000	0x00_FFFF_FFFF	DRAM	2GB	S/NS
0x08_8000_0000	0x09_FFFF_FFFF	DRAM	6GB	S/NS

Related information

[Command-line options](#) on page 18

4.4 Clock and timer

This section describes the frequencies of the clock and timer.

Cluster `clk_in` frequency parameter

100MHz.

GenericTimer `base_frequency` parameter

100MHz.

² The Foundation Model v2.1 only. Not the Foundation Platform.

³ The Foundation Platform uses the GICv3 memory map by default.

4.5 Interrupt maps

You can find information on the SPIs and PPIs on the GIC that the platform assigns.



Shared Peripheral Interrupt (SPI) and Private Peripheral Interrupt (PPI) numbers are mapped onto GIC interrupt IDs as the Arm® Generic Interrupt Controller Architecture Specification describes.

The following table lists the SPI assignments.

Table 4-5: Shared peripheral interrupt assignments

IRQ ID	SPI offset	Device
32	0	Watchdog, SP805
34	2	Dual-Timer 0, SP804
35	3	Dual-Timer 1, SP804
36	4	Realtime Clock, PL031
37	5	UART0, PL011
38	6	UART1, PL011
39	7	UART2, PL011
40	8	UART3, PL011
41	9	MCI, PL180, MCIINTRO
46	14	PL111 CLCD
47	15	Ethernet, SMSC 91C111
56	24	Trusted Watchdog, SP085
57	25	AP_REFCLK, Generic Timer, CNTPSIRQ
58	26	AP_REFCLK, Generic Timer, CNTPSIRQ1
59	27	EL2 Generic Watchdog WSO
60	28	EL2 Generic Watchdog WS1
74	42	Virtio block device
75	43	Virtio Plan 9
76	44	Virtio net device
92	60	cpu0 PMUIRQ
93	61	cpu1 PMUIRQ
94	62	cpu2 PMUIRQ
95	63	cpu3 PMUIRQ

The following table shows the PPI assignments:

Table 4-6: Private Peripheral Interrupt map

PPI	Device
3	Secure hypervisor virtual timer event

PPI	Device
4	Secure hypervisor physical timer event
9	Virtual maintenance interrupt
10	Hypervisor timer event
11	Virtual timer event
12	Hypervisor virtual timer event
13	Secure physical timer event
14	Non-secure physical timer event

4.6 System register block

The system register block provides a minimal set of registers.

This component only accepts word writes and aligned reads.

Table 4-7: System register block

Offset	Type	Bits	Register
0x0000	R/O	[31:0]	System ID Register
0x0004	R/W	[7:0]	User Programmable Switches
0x0008	R/W	[7:0]	LEDs
0x00A0	R/W	[31:0]	System configuration data
0x00A4	R/W	[31:0]	System configuration control
0x00A8	R/W	[31:0]	System configuration status

The System ID Register is divided into the following fields:

- ID[31:28] Revision.
0x2
Foundation Platform v9.1-v9.5.
0x3
Foundation Platform v9.6.
- ID[27:16] HBI board number.
0x010
Foundation Platform, default.
0x020
Arm® Base Platform FVP.
- ID[15:12] Build variant. The value depends on the following command-line options:
0x0
Variant A is the Foundation Platform with the GICv2 legacy map, when the `--no-givc3` command-line option is used.

0x1

Variant B is the Foundation Platform with the GICv3 64KB memory map, when the `--gicv3` command-line option is used. This is the default.

- ID[11:8] Platform type:

0x0

Board.

0x1

Model, default.

0x2

Emulator.

0x3

Simulator.

0x4

FPGA.

- ID[7:0] FPGA build.
 - Not used.

The user-programmable Switches store 8 bits of state that can be read or written by software on the platform. You can configure the startup value, `val`, using `--switches=val`.

The CLCD window shows the values of the switches and LEDs at runtime.

The system configuration control register provides two functions:

- Writing the value `0xC0800000` stops the simulation and returns control to the command line.
- Writing the value `0xC0900000` asserts and then clears the reset pins on all components in the simulation. It resets the system without clearing the contents of the RAMs.



Writes to the system configuration register can take several instructions to complete. Therefore, a write to this register must be followed by a `DSB` and infinite loop.

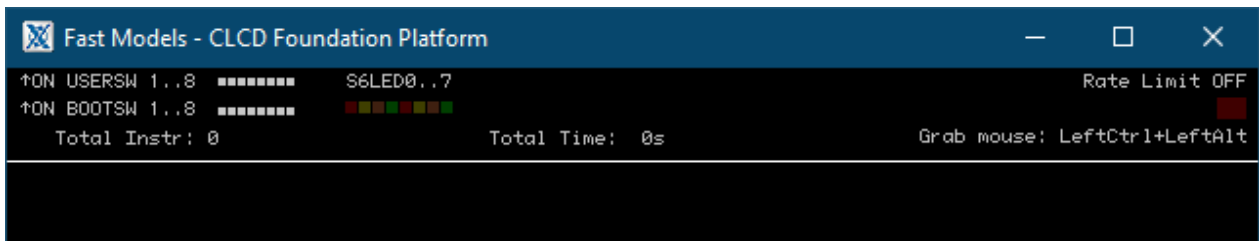
The system configuration data and status registers always return 0 on reads, and writes are ignored.

4.7 CLCD window

When the model starts, the Foundation Platform CLCD window opens, representing the contents of the simulated color LCD frame buffer. It automatically resizes to match the horizontal and vertical resolution set in the CLCD peripheral registers.

The following figure shows the Foundation Platform CLCD in its default state, immediately after being started:

Figure 4-1: CLCD window at startup



The top section of the CLCD window displays the following status information:

Total Instr

A counter showing the total number of instructions executed.

Total Time

A counter showing the total elapsed time, in seconds.

This is wall clock time, not simulated time.

Rate Limit

This option limits the rate of simulated time when the cores are in WFI, reset, or otherwise idle. Simulation time is restricted so that it more closely matches real time.

Rate Limit is disabled by default. Click the square button to enable it. The text changes from OFF to ON and the colored box becomes lighter red when the Rate Limit is enabled.



Note

You can also control whether the Rate Limit is enabled by using the `--(no-)rate-limit` parameter when instantiating the model.

Instr / sec

Shows the number of instructions executed per second of wall clock time.

Perf Index

The ratio of real time to simulation time. The larger the ratio, the faster the simulation runs. If you enable the Rate Limit feature, the Perf Index approaches unity.

Icons









Icons to represent different processor states.

The following table shows each of the possible icons:



The icons do not appear until you start the simulation.

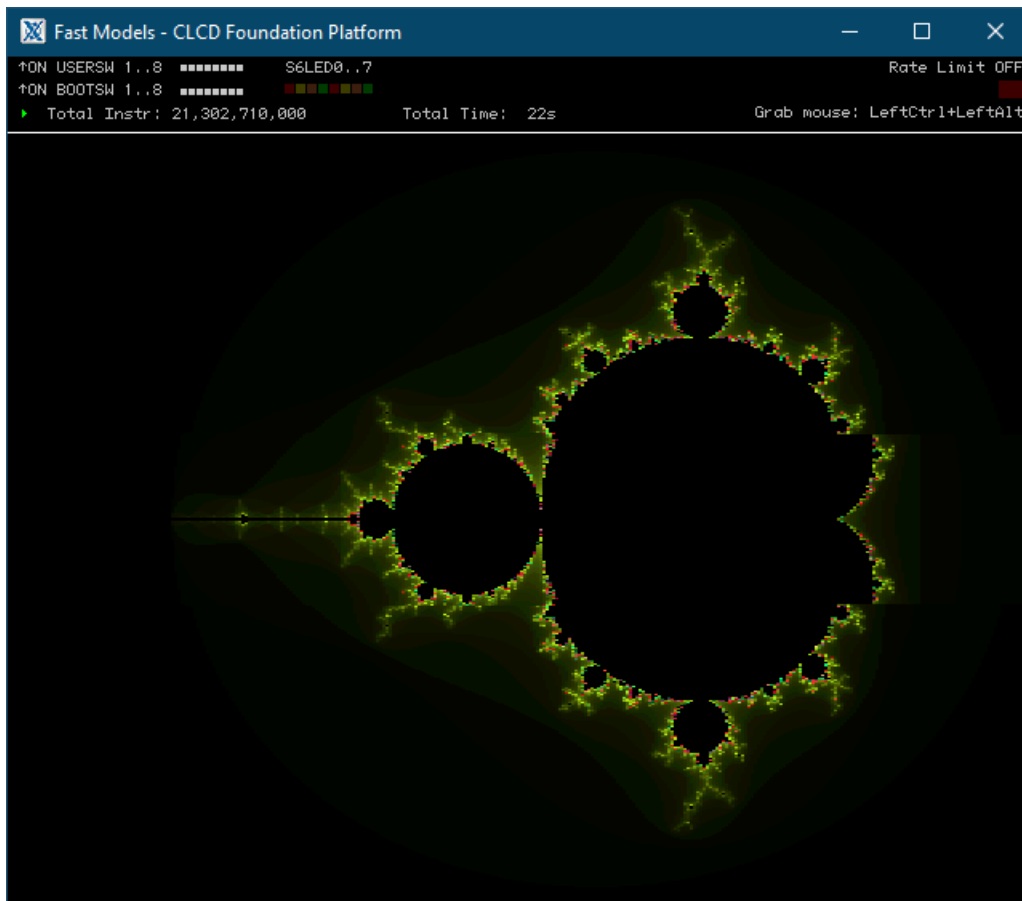
Table 4-8: Core run state icon descriptions

Icon	State label	Description
	UNKNOWN	Run status unknown, that is, simulation has not started.
	RUNNING	The core is running, is not idle, and is executing instructions.
	HALTED	An external halt signal is asserted.
	STANDBY_WFE	The last instruction executed was WFE, and standby mode has been entered.
	STANDBY_WFI	The last instruction executed was WFI and standby mode has been entered.
	IN_RESET	An external reset signal is asserted.
	DORMANT	Partial core power down.
	SHUTDOWN	Complete core power down.



The icons do not appear until you start the simulation.

The large area at the bottom of the window displays the contents of the CLCD buffer, for example:

Figure 4-2: CLCD window active

You can hide the host mouse pointer by pressing the **Left Ctrl+Left Alt** keys. Press the keys again to redisplay the host mouse pointer. Only the **Left Ctrl** key is operational. The **Ctrl** key on the right-hand side of the keyboard does not have the same effect.

4.8 UARTs

When the Foundation Platform starts, it initializes four UARTs. For each UART, it searches for a free TCP port to use for telnet access to the UART. It searches by sequentially scanning a range of 100 ports and using the first free port. The start port defaults to 5000 and you can change it using the `--uart-start-port` command-line parameter.

Connecting a terminal or program to the given port displays and receives output from the associated UART and permits input to the UART.

If no terminal or program is connected to the port when data is output from the UART, a terminal is started automatically.



A terminal only starts automatically if the `DISPLAY` environment variable is set and is not empty.

UART output

For the UART output to be visible, both `xterm` and `telnet` must be installed on the host, and be specified in your `PATH`.

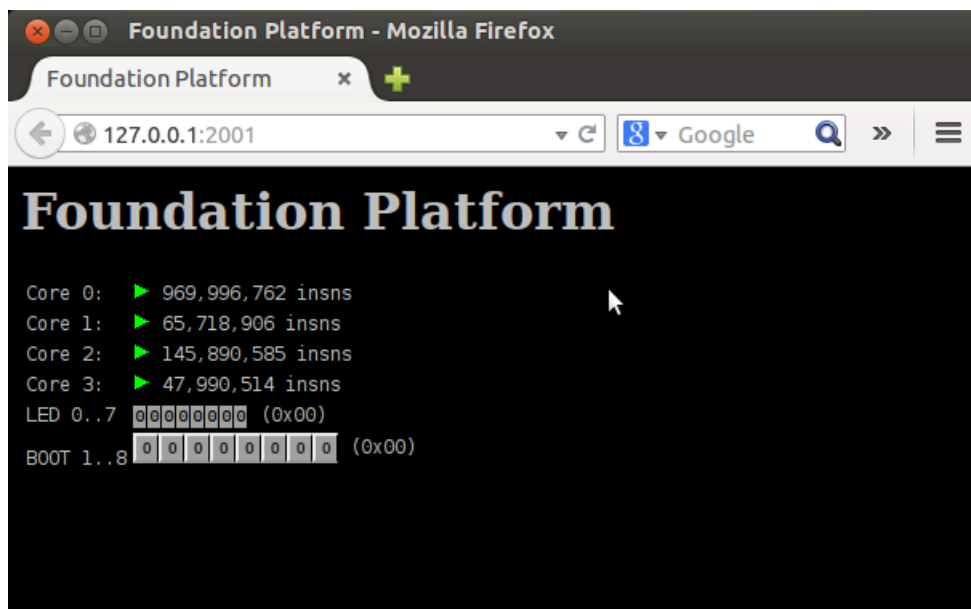
Alternatively you can redirect output from each of the four available UARTs to a file or to stdout, using the `--uartN-outfile=file` command-line parameter, where *N* is in the range 0-3. Specifying a filename of `-` redirects the output to stdout.

4.9 Multicore configuration

By default, the platform starts up with a single core that begins executing from the entry point in the last provided ELF image, or address 0 if no ELF images are provided.

You can configure the platform using `--cores=N` to have up to four processor cores. Each core starts executing the same set of images, starting at the same address. The `--visualization` command-line option which is used with the multicore option, results in a visualization window.

Figure 4-3: Multicore option with number of cores = 4



4.10 Semihosting

Semihosting enables code running on a platform model to directly access the I/O facilities of a host computer. To use semihosting, you must have connected the model to a debugger, for example Arm® Development Studio Debugger.

The simulator handles semihosting by either:

- Intercepting `svc 0x123456` or `0xAB` in AArch32 execution state, depending on whether the processor is in the Arm® or Thumb® instruction set state.
- Intercepting `HLT 0xF000` in AArch64 execution state.

Related information

[Semihosting for AArch32 and AArch64](#)

[Using semihosting to access resources on the host computer](#)