Arm®v8-A Foundation Platform

Version 11.14

User Guide

arm

100961_1114_00_en

Arm®v8-A Foundation Platform

User Guide

Release Information

Document History

Issue	Date	Confidentiality	Change
A	17 October 2012	Non-Confidential	First release.
B	01 May 2013	Non-Confidential	Minor updates. Directory structure changed.
C	13 November 2013	Non-Confidential	Memory maps added. Update for Foundation Model v2.
D	24 June 2014	Non-Confidential	Minor updates. Update for v2.1.
E	28 November 2014	Non-Confidential	Update for v9.1. Added virtiop9 control.
F	28 February 2015	Non-Confidential	Update for v9.2.
G	31 May 2015	Non-Confidential	Update for v9.3.
H	31 August 2015	Non-Confidential	Update for v9.4.
I	30 November 2015	Non-Confidential	Update for v9.5.
J	29 February 2016	Non-Confidential	Update for v9.6.
K	31 May 2016	Non-Confidential	Update for v10.0.
L	31 August 2016	Non-Confidential	Update for v10.1.
M	11 November 2016	Non-Confidential	Update for v10.2.
N	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.
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	05 September 2019	Non-Confidential	Update for v11.8.
1108-01	03 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.
1111-00	09 June 2020	Non-Confidential	Update for v11.11.
1112-00	22 September 2020	Non-Confidential	Update for v11.12.
1113-00	09 December 2020	Non-Confidential	Update for v11.13.
1114-00	17 March 2021	Non-Confidential	Update for v11.14.

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

developer.arm.com

Progressive terminology commitment

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

We believe that this document contains no offensive terms. If you find offensive terms in this document, please contact *terms@arm.com*.

Contents
Arm®v8-A Foundation Platform User Guide

Preface

This preface introduces the *Arm®v8-A Foundation Platform User Guide*.

It contains the following:

About this book

This document describes the Arm®v8-A Foundation Platform for the Armv8-A architecture. It is an aid for hardware and software developers in developing Armv8-A products.

Using this book

This book is organized into the following chapters:

Chapter 1 Arm®v8-A Foundation Platform Introduction

This chapter introduces the Armv8-A Foundation Platform.

Chapter 2 Getting Started

This chapter describes testing the Armv8-A Foundation Platform installation.

Chapter 3 Programming Reference

This chapter describes the Armv8-A Foundation Platform.

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 *Armv8-A Foundation Platform User Guide*.
- The number 100961_1114_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® Documentation*.
- *Technical Support*.
- *Arm® Glossary*.

Chapter 1
Arm®v8-A Foundation Platform Introduction

This chapter introduces the Armv8-A Foundation Platform.

It contains the following sections:

1.1 Platform introduction

The Armv8-A Foundation Platform is an enabling platform for the Armv8-A architecture.

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.

1.2 Software requirements

This section describes the host software that is required to run the Armv8-A Foundation Platform.

Operating Systems

Red Hat Enterprise Linux 6 or 7 (for 64-bit architectures), Ubuntu 16.04 or 18.04 Long Term Support (LTS).

Note

Currently, there is no support for running the platform on other operating systems. However, the platform runs on any recent x86 64-bit Linux OS provided glibc v2.3.2, or higher, and libstdc++ 6.0.0, or higher, are present.

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.

1.3 Platform overview

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

This section contains the following subsections:

1.3.1 Features of the Foundation Platform

The Armv8-A Foundation Platform provides the following features:

- An Armv8-A cluster model containing 1-4 cores.
- 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.

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 references

3.2 Foundation Platform memory map on page 3-29

Related information

Armv8-A Architecture Reference Manual

1.3.2 Processor models in the Foundation Platform

The processor models in this platform are not based on any existing processor design, but conform to the Armv8-A architectural specifications.

They implement:

- Armv8 versions 8.0-8.7.
- 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.

1.3.3 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.

1.3.4 Foundation Platform block diagram

This figure shows the block diagram for the Foundation Platform.

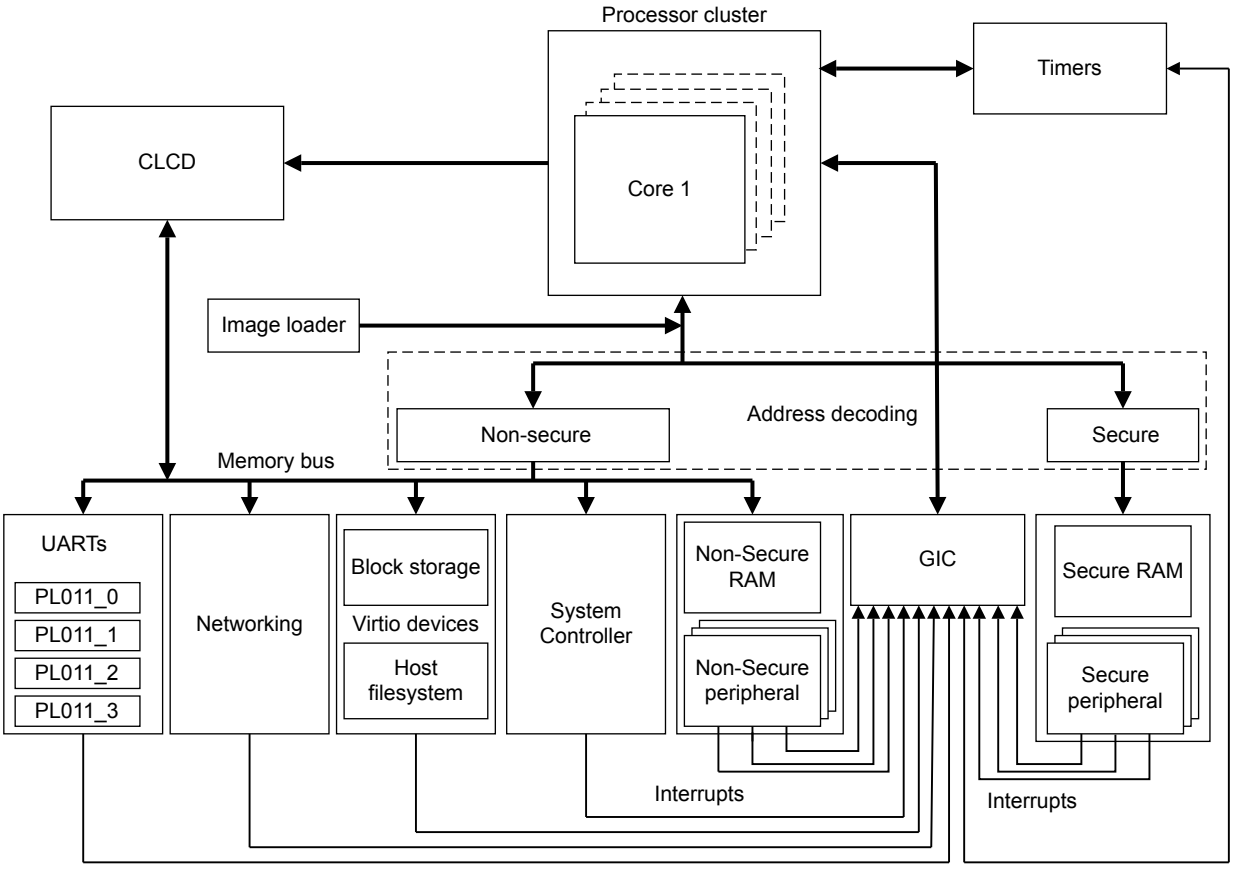

Figure 1-1 Block diagram of Armv8-A Foundation Platform

Note

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

1.3.5 Limitations of the Foundation Platform

The following restrictions apply to the Armv8-A 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.
- There is no support for Thumb®2EE.
- There is no support for the Armv8 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*.

1.4 Data collection in the Foundation Platform

Arm periodically collects anonymous information about the usage of our products to understand and analyze what components or features you are using, with the goal of improving our products and your experience with them. Product usage analytics contain information such as system information, settings, and usage of specific features of the product. They do not include any personal information.

Host information includes:

- Operating system name, version, and locale.
- Number of CPUs.
- Amount of physical memory.
- Screen resolution.
- Processor and GPU type.

Feature tracking information includes:

Table 1-1 Fast Models analytics data points

Name	Description	Since
Platform name	• Tracked: — Name of the platform model being run — Fast Models version and build number that was used to build the platform model. — Whether the platform model is a standalone product or was supplied as part of the Fast Models product. [a] • Reported: Percentage of users using the different platforms. • Data type: Text. • Send policy: Every invocation. • Trigger points: On starting the simulation.	v11.8
Session length	• Tracked: Length of time the platform was used. • Reported: Average time the different platforms are used. • Data type: Text. • Send policy: Every invocation. • Trigger points: On exiting the simulation.	v11.9
Debug server	• Tracked: Whether a CADI or Iris debugger was connected. • Reported: Type of debug server that was started, either CADI or Iris. • Data type: Text. • Send policy: Every invocation. • Trigger points: Debug server startup.	v11.10
Arm IP	• Tracked: Names of Fast Models core, System IP, and GPU components that are included in the simulation. • Reported: Component names, for example DP500 or SMMUv3AEM. These are not hierarchical names. • Data type: Text. • Send policy: Every invocation. • Trigger points: Component instantiation.	v11.10

[a] This feature is tracked since v11.10.

Note

- Analytics gathering is enabled by default. Use the `--disable-analytics` command-line option to disable it for the current invocation, or set the `ARM_DISABLE_ANALYTICS` environment variable to a non-zero value to disable it for all invocations.
- Querying the available options using `--help` does not trigger reporting.
- The names of non-Arm platforms or components are obfuscated.

Chapter 2
Getting Started

This chapter describes testing the Armv8-A Foundation Platform installation.

It contains the following sections:

2.1 Verifying the installation

The Foundation Platform is available only as a prebuilt platform binary.

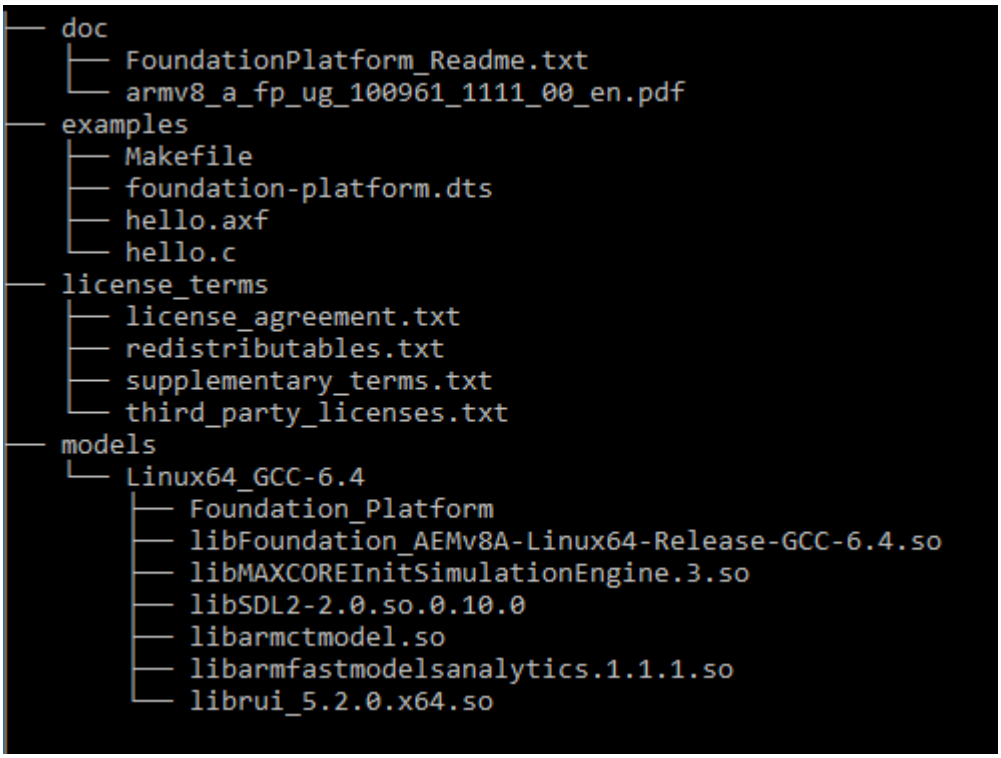

Figure 2-1 Contents of the installation

The contents of the installation are:

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.

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

models
Contains the Foundation Platform executable and libraries that are required at run time:

`Foundation_Platform`
The Armv8-A Foundation Platform executable.

`libFoundation_AEMv8A-Linux64-Release-GCC-6.4.so`
EVS shared library. This library is required because the Foundation Platform is an EVS platform.

`libMAXCOREInitSimulationEngine.3.so`
Required for initializing the platform.

`libSDL2-2.0.so.0.10.0`
Simple DirectMedia Layer library, required for CLCD visualization.

`libarmctmodel.so`
Code translation library.

`libarmfastmodelsanalytics.1.1.1.so` and `librui_5.2.0.x64.so`
Required if analytics gathering is enabled.

Related tasks

2.2 Running the example program on page 2-21

Related information

Device Tree

2.2 Running the example program

You can use the example program that is supplied to confirm that the Armv8-A Foundation Platform is working correctly.

Procedure

1. Run the example from the command line: `./Foundation_Platform --image hello.axf`
2. Optionally add `--quiet` to suppress all output except for the output from the example program.
   Results:

   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!

Info: /OSCI/SystemC: Simulation stopped by user.
```

   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.

2.3 Troubleshooting the example program

You can 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` is too old, or your `libstdc++` is too old, you get the following messages:

```
./Foundation_Platform: /usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4' not found (required by
Foundation_Platform)
./Foundation_Platform: /lib64/libc.so.6: version `GLIBC_2.3.2' not found (required by Foundation_Platform)
./Foundation_Platform: /lib64/libc.so.6: version `GLIBC_2.2.5' not found (required by Foundation_Platform)
```

`libstdc++` and `glibc` are normally part of your core OS installation.

2.4 Using Linux

Arm provides validated Linux and Android deliverables for the Armv8-A Foundation Platform.

These are available on the Arm Community website at *Arm Development Platforms*. To get started with Linux on the Armv8-A Foundation Platform, see *Armv8-A FVPs* on Arm Community.

Chapter 3
Programming Reference

This chapter describes the Armv8-A Foundation Platform.

It contains the following sections:

3.1 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 3-1 Command-line options

Option	Description
`--arm-v8.`*n*	Enable the Armv8.*n* version of the architecture, where 0 <= *n* <= 7. The default is `--arm-v8.7`.
`--bigendian`	Start processors in big-endian mode. The default is little-endian.
`--block-device=`*file*	Image file to use as persistent block storage.
`--cadi-server`	Start the CADI server. This option allows a CADI-enabled debugger to connect to targets in the simulation.
`--cores=`*N*	Specify the number of processors, where *N* is in the range 1-4. The default is 1.
`--(ns)data=`*file@address*	Raw file to load at an address in non-secure or secure memory. Use `--nsdata` to load data into non-secure memory when the `--secure-memory` option is enabled.
`--disable-analytics`	Disable product analytics gathering for the current invocation. Set the `ARM_DISABLE_ANALYTICS` environment variable to a non-zero value to disable it for all invocations.
`--(no-)gicv3`	Enable GICv3 or the legacy, compatible GICv2. The default is `--gicv3`.
`--help`	Display the command-line options and quit.
`--image=`*file*	ELF image to load.
`--min-sync-latency=`*N*	Number of ticks to simulate before synchronizing. Events that occur at a higher frequency than this value are missed. The default is 100.
`--network=(none|nat|bridged)`	Configure the network access mode. The default is none.
`--network-bridge=`*dev*	Bridged network device name. The default is ARM0.
`--network-mac-address`	MAC address to use for networking. The default is `00:02:f7:ef:f6:74`.
`--network-nat-ports=`*M*	Optional comma-separated list of NAT port mappings in the form: *host_port=model_port*, for example, `8022=22`.
`--network-nat-subnet=`*S*	Subnet used for NAT networking. The default is `172.20.51.0/24`.
`--p9_root_dir=`*dir*	Host folder to be shared between the host and the guest.
`--print-port-number`	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.
`--quiet`	Suppress any non-simulated output on `stdout` or `stderr`.
`--quantum=`*N*	Number of ticks to simulate for each quantum. The default is 10000.
`--(no-)rate-limit`	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.
`--read-only`	Mount block device image in read-only mode.

Table 3-1 Command-line options (continued)

Option	Description
`--(no-)secure-memory`	Enable or disable separate secure and non-secure address spaces. The default is disabled.
`--(no-)semihost`	Enable or disable semihosting support. The default is enabled.
`--semihost-cmd=`*`cmd`*	A string that is used as the semihosting command line.
`--semihosting-heap_base=`*`address`*	Virtual address of the heap base. The default is 0.
`--semihosting-heap_limit=`*`address`*	Virtual address of the top of the heap. The default is `0xFF000000`.
`--semihosting-stack_base=`*`address`*	Virtual address of the base of the descending stack. The default is `0xFFFF0000`.
`--semihosting-stack_limit=`*`address`*	Virtual address of the stack limit. The default is `0xFF000000`.
`--(no-)sve`	Enable or disable *Scalable Vector Extension* (SVE) and SVE2. The default is enabled. This option requires Armv8.2 or a later architecture to be enabled. If SVE is enabled, the options described in *Table 3-2 Additional SVE-related options* on page 3-27 can take effect.
`--switches=`*`val`*	Initial setting of switches in the system register block. The default is 0.
`--trace=`*`file`*	Enable the TarmacTrace extension. 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.
`--uart`*`N`*`-outfile=`*`file`*	Redirect output from UART*N* to a file, where *N* is in the range 0-3. Specify a filename of `-` to redirect output to stdout. If no filename is specified, the option is ignored.
`--uart-start-port=`*`P`*	Attempt to listen on a free TCP port in the range *P* to *P*+100 for each UART. The default is 5000.
`--use-real-time`	Sets the generic timer registers to report a view of real time as it is seen on the host platform, instead of simulated time.
`--version`	Display the version and build numbers and quit.
`--(no-)visualization`	Starts a small web server to visualize the platform state. The default is disabled.

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.

The following options only take effect if the Scalable Vector Extension (SVE) is enabled which is controlled by the `--(no-)sve` option:

Note

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

Table 3-2 Additional SVE-related options

Option	Type	Description
`--clear-constrained-lanes`	`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.
`--combine-movprfx-and-destructive`	`bool`	Attempt to combine the execution of `MOVPRFX` and the destructively encoded instruction that follows it. The default is false.
`--disable-speculative-accesses`	`bool`	All speculative memory accesses behave as though faulting without accessing memory. The default is false.
`--enable-at-reset`	`bool`	Start with system registers set up for Scalable Vector Extension use. The default is false.
`--ffr-16b-pattern-UNKNOWN`	`int`	A specific 16-bit UNKNOWN value that is used by parameter `force_UNKNOWN_to_ffr`. The default is 0.
`--force-UNKNOWN-to-ffr`	`int`	Governs the behavior if `WRFFR` writes a non-monotonic value to `FFR`. Possible values are: `0x0` Write non-canonical value to `FFR`. This is the default. `0x1` Overwrite `FFR` with a specific 16-bit UNKNOWN value. See `ffr_16b_pattern_UNKNOWN`. `0x2` Clear all bits above first zero.
`--fp-exception-report-lowest`	`bool`	If true, for multiple trapped FP exceptions, report the lowest lane in `VECITR`. Otherwise, report the highest. The default is false.
`--fp-exception-set-tfv`	`bool`	Set `ESR_ELx.TFV` during FP exception. Trapped exception flags are valid. The default is true.
`--fp-exception-set-vecitr`	`bool`	If true, set `ESR_ELx.VECITR` during FP exception. Otherwise, set RES0. The default is false.
`--has-sve2`	`bool`	Whether SVE2 is implemented. The default is false.
`--has-sve2-aes`	`int`	If SVE2 is implemented, whether SVE2 AES instructions are implemented. Possible values are: `0x0` Not implemented. `0x1` SVE2 `AESE`, `AESD`, `AESMC`, and `AESIMC` are implemented. `0x2` Same as 1, but in addition, SVE2 `PMULLB` and `PMULLT` with 64-bit source are implemented. This is the default.
`--has-sve2-bit-perm`	`bool`	If SVE2 is implemented, whether BitPerm instructions are implemented. The default is true.
`--has-sve2-sha3`	`bool`	If SVE2 is implemented, whether SHA3 instructions are implemented. The default is true.
`--has-sve2-sm4`	`bool`	If SVE2 is implemented, whether SM4 instructions are implemented. The default is true.
`--has-sve-bf16`	`bool`	Whether SVE BFloat16 instructions are implemented. The default is true.
`--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.

Table 3-2 Additional SVE-related options (continued)

Option	Type	Description
`--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 `0xdeaddeaddeaddead`.
`--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.

Related concepts

3.9 Multicore configuration on page 3-42
3.7 Web interface on page 3-40
3.10 Semihosting on page 3-43

3.2 Foundation Platform memory map

This section describes the memory map for the Armv8-A Foundation Platform.

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

Table 3-3 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 Armv8-A Foundation Platform. This map is based on the Versatile Express RS2 memory map with extensions.

Note

- 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. This rule is applicable to Foundation Model v2 and Foundation Platform v9.
- Writes are ignored.
- Accesses from Foundation Model v1 cause an abort exception.

Note

The Security column in the following table applies to the Foundation Model v2 and Foundation Platform v9 only.

Table 3-4 Armv8-A Foundation Platform memory map

Start address	End address	Foundation v1 peripheral	Foundation v2 and v9 peripherals	Size	Security (v2 and v9 only)
`0x00_0000_0000`	`0x00_03FF_FFFF`	RAM	Trusted Boot ROM, secureflash	64MB	S
`0x00_0400_0000`	`0x00_0403_FFFF`	RAM	Trusted SRAM	256KB	S
`0x00_0600_0000`	`0x00_07FF_FFFF`	RAM	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 [b]	-
`0x00_1A00_0000`	`0x00_1AFF_FFFF`	Ethernet, SMSC 91C111	Ethernet, SMSC 91C111	16MB	S/NS
`0x00_1C01_0000`	`0x00_1C01_FFFF`	System Registers	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	UART0, PL011	64KB	S/NS
`0x00_1C0A_0000`	`0x00_1C0A_FFFF`	UART1, PL011	UART1, PL011	64KB	S/NS

[b] 8MB of VRAM is replicated 4 times in memory.

Table 3-4 Armv8-A Foundation Platform memory map (continued)

Start address	End address	Foundation v1 peripheral	Foundation v2 and v9 peripherals	Size	Security (v2 and v9 only)
0x00_1C0B_0000	0x00_1C0B_FFFF	UART2, PL011	UART2, PL011	64KB	S/NS
0x00_1C0C_0000	0x00_1C0C_FFFF	UART3, PL011	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	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
0x00_2C00_0000	0x00_2C00_1FFF	-	GIC Physical CPU interface, GICC[c]	8KB	S/NS
0x00_2C00_1000	0x00_2C00_1FFF	GIC Distributor	GIC Distributor[d]	4KB	-
0x00_2C00_2000	0x00_2C00_2FFF	GIC Processor Interface	GIC Processor Interface[d]	4KB	-
0x00_2C00_4000	0x00_2C00_4FFF	GIC Processor Hyp Interface	GIC Processor Hyp Interface[d]	4KB	-

[c] The Foundation Model v2.1 only. Not the Foundation Platform.
[d] The Foundation Platform uses the GICv3 memory map by default.

Table 3-4 Armv8-A Foundation Platform memory map (continued)

Start address	End address	Foundation v1 peripheral	Foundation v2 and v9 peripherals	Size	Security (v2 and v9 only)
0x00_2C00_5000	0x00_2C00_5FFF	GIC Hyp Interface	GIC Hyp Interface[d]	4KB	-
0x00_2C00_6000	0x00_2C00_7FFF	GIC Virtual CPU Interface	GIC Virtual CPU Interface[d]	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[c]	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 (0GB - 2GB)	DRAM (0GB - 2GB)	2GB	S/NS
0x08_8000_0000	0x09_FFFF_FFFF	DRAM (2GB - 8GB)	DRAM (2GB - 8GB)	6GB	S/NS

Related references

3.1 Command-line options on page 3-25

3.3 Clock and timer

This section describes the frequencies of the clock and timer.

Cluster `clk_in` frequency parameter
100MHz.

GenericTimer `base_frequency` parameter
100MHz.

3.4 Interrupt maps

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

Note

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 3-5 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, MCIINTR0
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 WS0
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 3-6 Private Peripheral Interrupt map

PPI	Device
3	Secure hypervisor virtual timer event
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

3.5 System register block

The system register block provides a minimal set of registers.

This component only accepts word writes and aligned reads.

Table 3-7 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
  Armv8-A 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 System ID register is not implemented in the Foundation Model v1. All unimplemented registers in the Foundation Model v1 system register block return the value `0xDEADDEAD` on reads. You can use this value to distinguish Foundation Model v1 from both Foundation Model v2 and Foundation Platform v9, and FVP VE Base Platform.

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`*.

You can view and set the switches at run time from the web interface.

The LEDs store 8 bits of state that software can read or write on the platform and can be viewed at runtime from the web interface.

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.

Note

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.

Related concepts

3.7 Web interface on page 3-40

3.6 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:

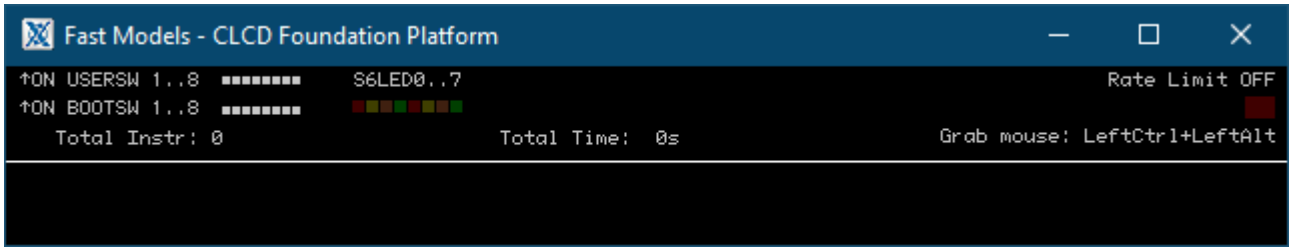

Figure 3-1 CLCD window at startup

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:

——— **Note** ———

The icons do not appear until you start the simulation.

———————————

Table 3-8 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.

Note

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:

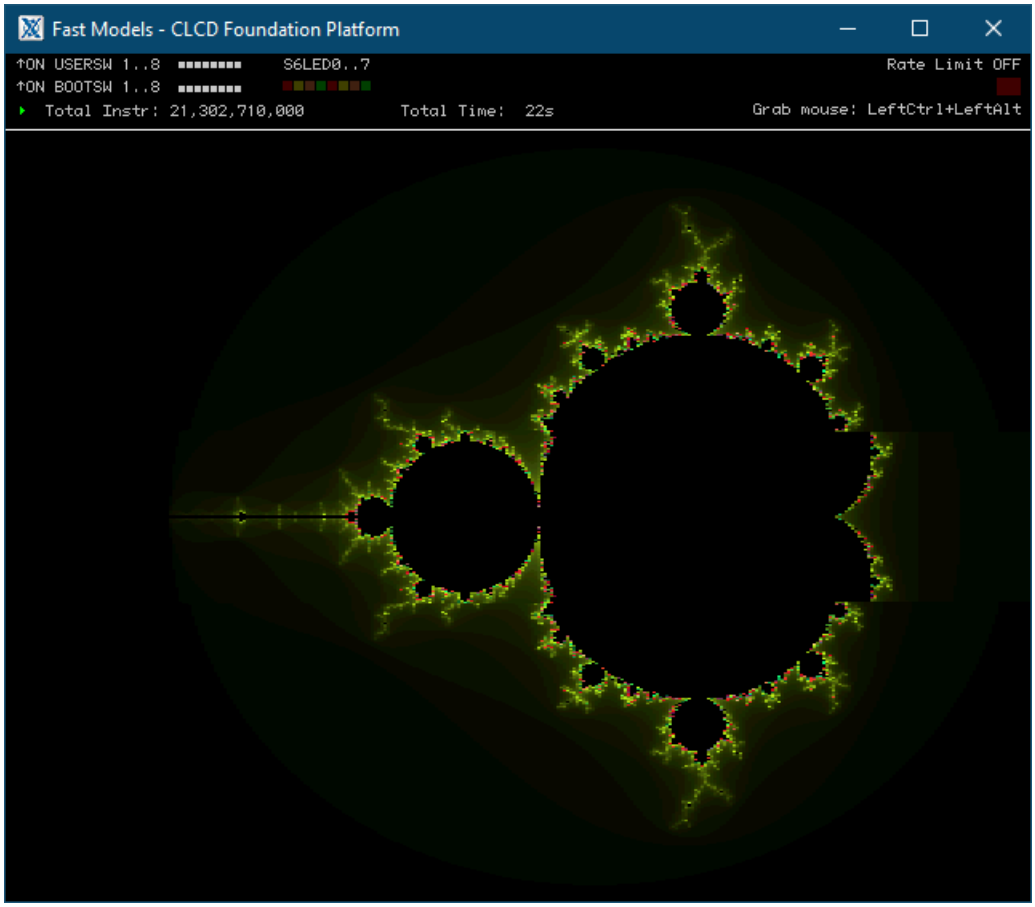

Figure 3-2 CLCD window active

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.

3.7 Web interface

This section describes the syntax to use on the command line.

You can use one of the following options on the command line:

- `./Foundation_Platform --visualization`
- `./Foundation_Platform --no-visualization`

Running the platform with the `--visualization` option, and without the `--quiet` option, shows the additional output:

```
terminal_0: Listening for serial connection on port 5000
terminal_1: Listening for serial connection on port 5001
terminal_2: Listening for serial connection on port 5002
terminal_3: Listening for serial connection on port 5003
Visualization web server started on port 2001
```

The `terminal_n` lines relate to the UARTs.

Go to the address http://127.0.0.1:2001 with your web browser.

The browser displays a visualization window.

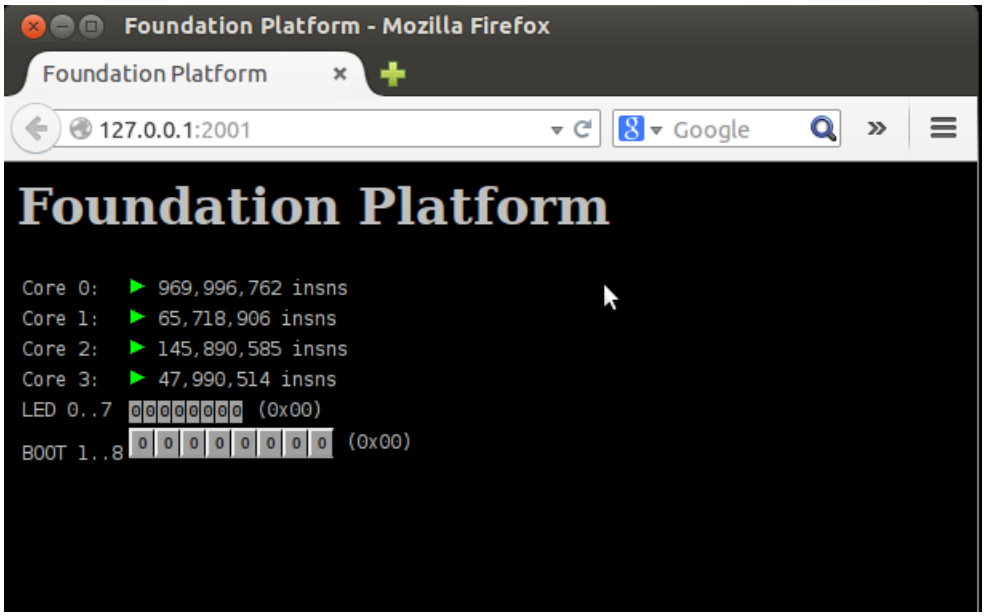

Figure 3-3 Visualization window

The visualization window provides a dynamic view of the state of various parts of the platform and the ability to change the state of platform switches.

Related concepts

3.8 UARTs on page 3-41

3.8 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.

Note

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.

3.9 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 3-4 Multicore option with number of cores = 4

3.10 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