# ARMv8-A Foundation Platform

**Version 11.0**

## User Guide

ARM


ARM 100961_1100_00_en

# ARMv8-A Foundation Platform

## User Guide



### Release Information

### Document History

| Issue | Date | Confidentiality | Change |
|---|---|---|---|
| A | 10 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. |

### Non-Confidential Proprietary Notice

not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

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



ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

### Confidentiality Status

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

Unrestricted Access is an ARM internal classification.

### Product Status

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

### Web Address

*http://www.arm.com*

# Contents
# ARMv8-A Foundation Platform User Guide

# Preface

This preface introduces the *ARMv8-A Foundation Platform User Guide*.

It contains the following:

# About this book

This document describes the ARMv8-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 Introduction*

This chapter describes the ARMv8-A Foundation Platform for the ARMv8-A architecture.

### *Chapter 2 Getting Started*

This chapter describes validation and testing on the ARMv8-A Foundation Platform.

### *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.

`mono`space
: 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 ARM 100961_1100_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

**Note**

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

## Other information

- *ARM Developer*.
- *ARM Information Center*.
- *ARM Technical Support Knowledge Articles*.
- *Support and Maintenance*.
- *ARM Glossary*.

# Chapter 1
# **Introduction**

This chapter describes the ARMv8-A Foundation Platform for the ARMv8-A architecture.

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 processors in this platform are not based on any existing processor design, but conform to the ARMv8-A architectural specifications. This means that you can use the platform for confirming software functionality, but you must not rely on the accuracy of cycle counts, low-level component interactions, or other hardware-specific behavior.

### Related concepts

*3.7 Web interface* on page 3-34.

## 1.2 ARMv8 64-bit architecture overview

The ARMv8 architecture is both an extension and a successor to the ARMv7 architecture.

ARMv8 introduces two execution states:

- A 32-bit execution state called AArch32.
- A 64-bit execution state called AArch64.

AArch32 state is compatible with the ARMv7-A architecture. Code executing in AArch32 state can only use the A32 and T32 instruction sets, although in ARMv8, these instruction sets have some new instructions relative to ARMv7.

The AArch64 state introduces a new fixed-length 32-bit instruction set called A64, while maintaining support for the same architectural capabilities as ARMv7-A, such as TrustZone® and Virtualization. Code executing in AArch64 state can only use the A64 instruction set.

AArch64 has four Exception levels, EL0-EL3, that replace the eight processor modes in ARMv7-A.

The least privileged, EL0, is equivalent to user-mode.

EL1 is equivalent to kernel-mode.

EL2 is used for hypervisors.

The highest privilege level, EL3, is used for the TrustZone security monitor.

Like ARMv7-A, AArch32 includes 13 general registers, R0-12, the Program Counter, R15, and two banked registers that contain the Stack Pointer, R13, and Link Register, R14. The User and System modes share these 16 registers and a *Program Status Register* (PSR). The new general purpose registers are all 64-bits wide to handle larger addresses, so 32-bit accesses use the lower halves of registers and either ignore or zero out the upper halves. The AArch32 registers map onto the lower halves of the AArch64 registers, and this permits AArch32 exceptions to be taken in AArch64 at a higher Exception level.

The two forms of instruction operate on either 32-bit or 64-bit values within the 64-bit general-purpose register file. Where a 32-bit instruction form is selected, the following holds true:

- The upper 32 bits of the source registers are ignored.
- The upper 32 bits of the destination registers are set to zero.
- Condition flags, where set by the instruction, are computed from the lower 32 bits.

For more information about the ARMv8-A architecture, see the *ARM® Architecture Reference Manual*.

## 1.3 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 14.04 Long Term Support (LTS), Ubuntu 16.04 LTS.
- Microsoft Windows 7 64-bit RTM or Service Pack 1, Professional, or Enterprise editions, Microsoft Windows 10 64-bit.

——— **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.

## 1.4 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:



### 1.4.1 Features and network support of the Foundation Platform

The ARMv8-A Foundation Platform has numerous features and two types of network support.

The platform provides:

- An ARMv8-A cluster model containing 1-4 cores that implements:
  - ARMv8.0, ARMv8.1, ARMv8.2, and ARMv8.3.
  - AArch64 at all Exception levels.
  - AArch32 support at EL0 and EL1.
  - Little and big endian at all Exception levels.
  - Generic timers.
  - Self-hosted debug.
  - GICv2, and optional GICv3 memory-mapped processor interfaces and distributor.
- 8GB of RAM.

  **Note**

  The platform simulates 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 Plan 9 filesystem.
- A CLCD that allows GUI visualization.
- Debug capabilities through the use of a CADI server.
- Support for the ARMv8-A *Scalable Vector Extension* (SVE).

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 the ARM Fast Models.

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

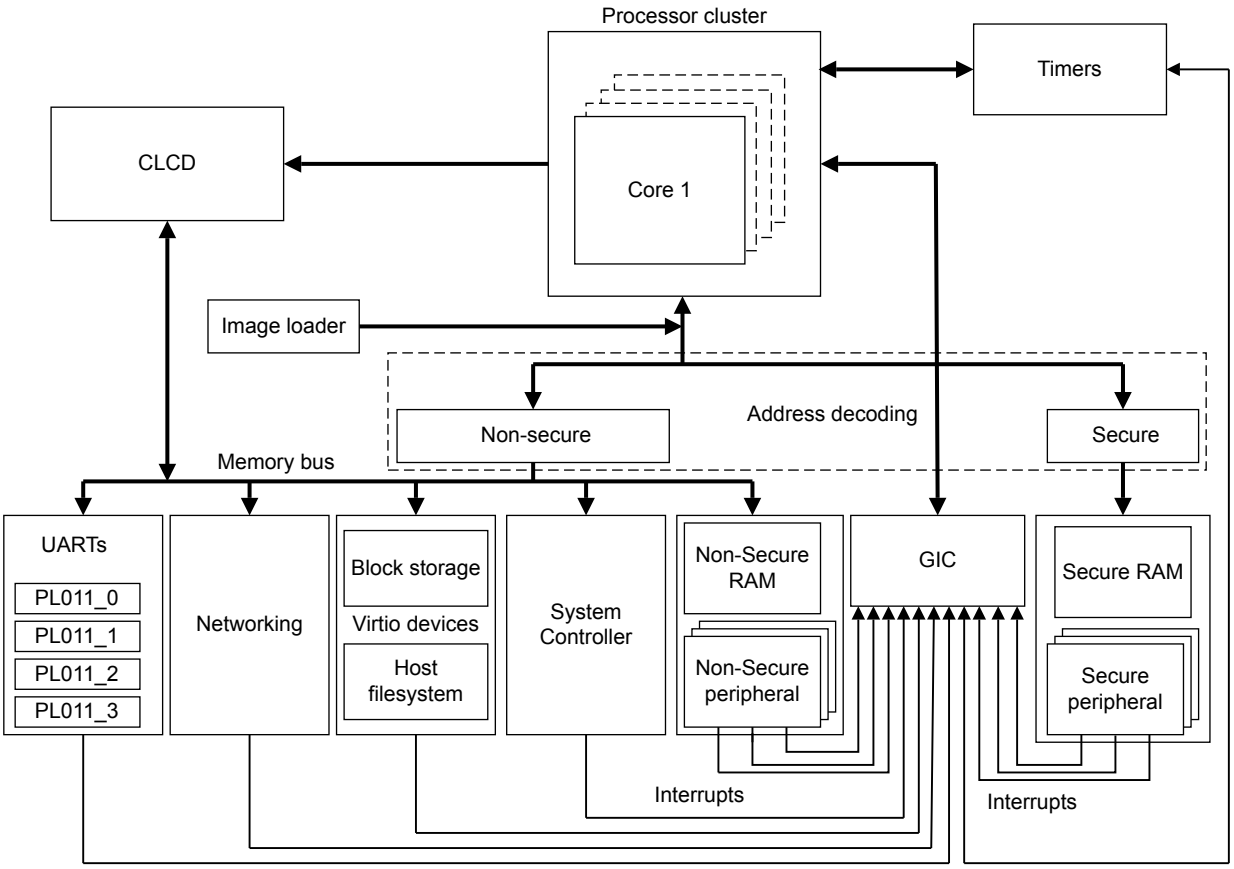


**Figure 1-1 Block diagram of ARMv8-A Foundation Platform**

**Note**

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

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.

The ARMv8-A Foundation Platform 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 the *Fast Models User Guide*.

### Related references

*3.2 ARMv8-A Foundation Platform memory map* on page 3-23.

## 1.4.2 Limitations of the Foundation Platform

There are some restrictions that 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.
- There is no Trace or other plug-in support.
- 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*.

# Chapter 2
# **Getting Started**

This chapter describes validation and testing on the ARMv8-A Foundation Platform.

It contains the following sections:

## 2.1 Verifying the installation

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

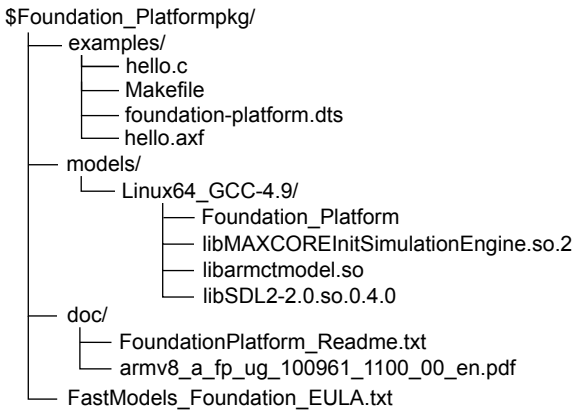


**Figure 2-1 Hierarchy of installed files**

The installation directory contents are:

`examples`
: Includes a C version and `.axf` file of the Hello World program. It also includes the Makefile and the example source code for the device tree, `foundation-platform.dts`.

`Foundation_Platform`
: The ARMv8-A Foundation Platform executable file.

`libMAXCOREInitSimulationEngine.so.2`
: Helper library required by the platform.

`libarmctmodel.so`
: Code translation library.

`libSDL2-2.0.so.0.4.0`
: Simple DirectMedia Layer library, required for CLCD visualization.

`FoundationPlatform_Readme.txt`
: Short summary of this user guide.

`armv8_a_fp_ug_100961_<revision>_00_en.pdf`
: This document.

`FastModels_Foundation_EULA.txt`
: End-user license agreement.

### Related tasks

*2.2.1 Running the example program* on page 2-18.

## 2.2 The example program

You can use the example program that is supplied to confirm that the ARMv8-A Foundation Platform is working correctly.

This section contains the following subsections:



### 2.2.1 Running the example program

This section describes how to run the example program.

Run the platform with the following command line:

#### Procedure

1. `./Foundation_Platform --image hello.axf`
2. Add `--quiet` to suppress everything except for the output from the example program.
   You receive a similar output to the following:

```
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
Simulation is started
Hello, 64-bit world!
Simulation is terminating. Reason: Simulation stopped
```

   The example demonstrates that the platform initializes correctly as it loads and executes the example program. It also demonstrates that the semihosting calls to print output and stop the platform work properly.

### 2.2.2 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.

## 2.3 Using Linux

You can also use Linux with the ARMv8-A Foundation Platform.

For information on configuring and building the arm64 port of Linux to run on the ARMv8-A Foundation Platform, see the Linaro website at *http://www.linaro.org/engineering/armv8*.

# 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. Run the platform with `--help` to see a summary of the available commands.

The syntax to use on the command line is:

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

**Table 3-1 Command-line options**

| | |
|---|---|
| `--arm-v8.`*n* | Enable the ARMv8.*n* version of the architecture, where 0 <= *n* <= 3. The default is `--arm-v8.3`. |
| `--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 debuggers 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. |
| `--(no-)gicv3` | Enable GICv3 or the legacy, compatible GICv2. The default is `--no-gicv3`, that is, GICv2. |
| `--help` | Display the command-line options and quit. |
| `--image=`*file* | ELF image to load. |
| `--network=(none\|nat\|bridged)` | Configure mode of network access. 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. |
| `--quiet` | Suppress any non-simulated output on `stdout` or `stderr`. |
| `--(no-)rate-limit` | Restrict 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. |
| `--(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. |
| `--(no-)sve` | Enable or disable *Scalable Vector Extension* (SVE). The default is enabled. This option requires ARMv8.2 or a later architecture to be enabled. |
| `--switches=`*val* | Initial setting of switches in the system register block. The default is 0. |
| `--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. The generic timer registers are irrespective of how slow or fast the simulation runs. |

| | |
|---|---|
| `--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.

### Related concepts

*3.9 Multicore configuration* on page 3-36.
*3.7 Web interface* on page 3-34.
*3.10.1 Semihosting* on page 3-37.

## 3.2 ARMv8-A 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 3-2 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 3-3 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 [a] | - |
| `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 |

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

**Table 3-3 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_1C16_FFFF | - | Virtio Plan 9 for v9, Warning + RAZ/W for v2.1 | - | - |
| 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[b] | 8KB | S/NS |
| 0x00_2C00_1000 | 0x00_2C00_1FFF | GIC Distributor | GIC Distributor[c] | 4KB | - |
| 0x00_2C00_2000 | 0x00_2C00_2FFF | GIC Processor Interface | GIC Processor Interface[c] | 4KB | - |
| 0x00_2C00_4000 | 0x00_2C00_4FFF | GIC Processor Hyp Interface | GIC Processor Hyp Interface[c] | 4KB | - |
| 0x00_2C00_5000 | 0x00_2C00_5FFF | GIC Hyp Interface | GIC Hyp Interface[c] | 4KB | - |

[b] The Foundation Model v2.1 only. Not the Foundation Platform.
[c] The Foundation Platform uses the GICv2 memory map by default, or if you use the `--no-gicv3` configuration parameter.

**Table 3-3 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_6000 | 0x00_2C00_7FFF | GIC Virtual CPU Interface | GIC Virtual CPU Interface[c] | 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[b] | 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 3-21.

## 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 3-4 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 |
| 92 | 60 | cpu0 PMUIRQ |
| 93 | 61 | cpu1 PMUIRQ |
| 94 | 62 | cpu2 PMUIRQ |
| 95 | 63 | cpu3 PMUIRQ |

The following table shows the PPI assignments:

**Table 3-5 Private Peripheral Interrupt map**

| PPI | Device |
|---|---|
| 9 | Virtual maintenance interrupt |
| 10 | Hypervisor timer event |
| 11 | 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 3-6 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. This is the default.

  `0x1`
  : Variant B is the Foundation Platform with the GICv3 64kB memory map, when the `--gicv3` command-line option is used.
- 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 3-34.

## 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:

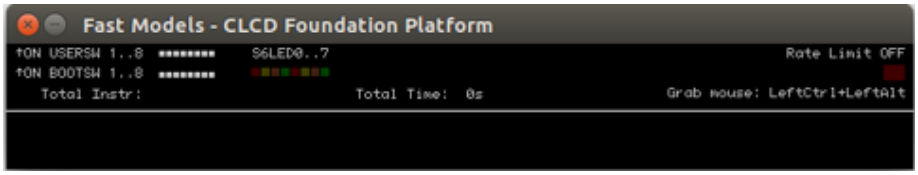


**Figure 3-1 CLCD window at startup**

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 3-7 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. |

**Table 3-7 Core run state icon descriptions (continued)**

| Icon | State label | Description |
|---|---|---|
|  | 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. The following figure shows this:

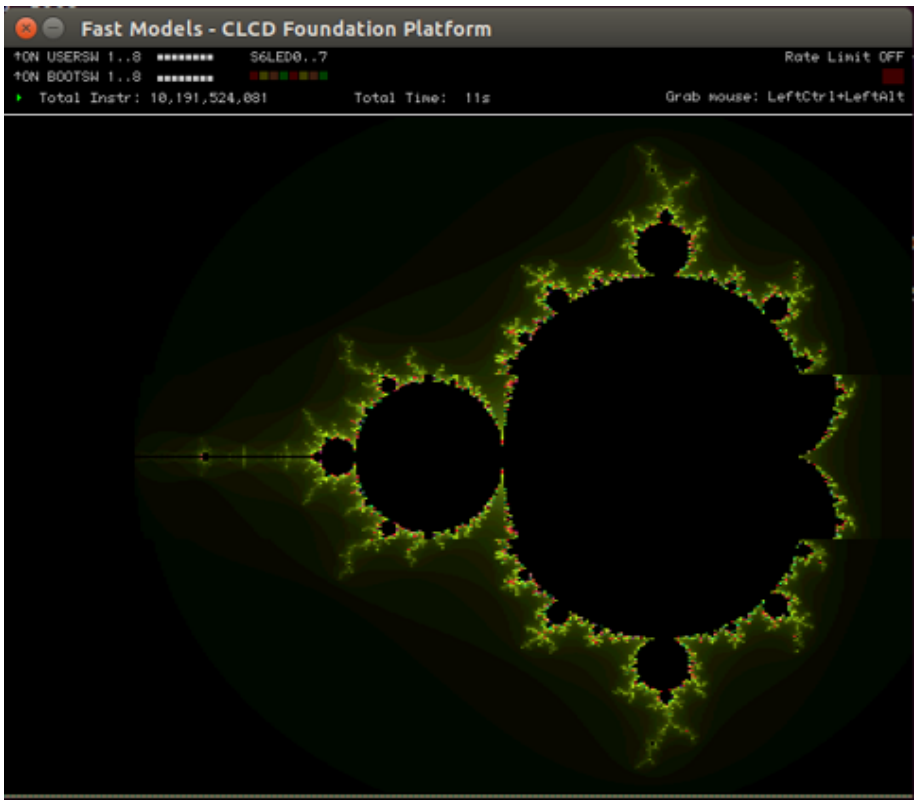


**Figure 3-2 CLCD window active**

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 in 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.

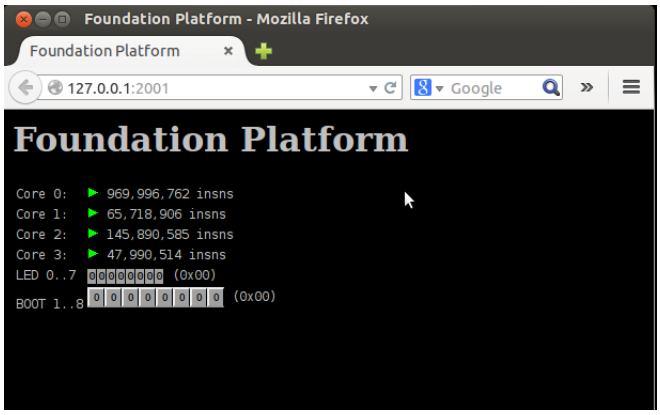


**Figure 3-3 Visualization window**

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 3-35.

## 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 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.

## 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.

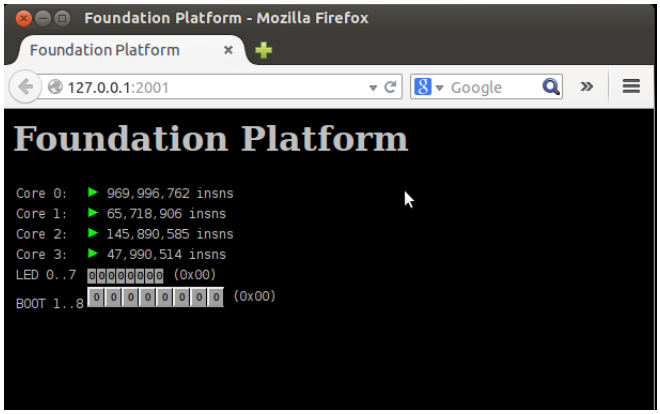


**Figure 3-4 Multicore option with number of cores = 4**

## 3.10 Semihosting overview

This section describes semihosting and semihosting configuration.

This section contains the following subsections:



### 3.10.1 Semihosting

Semihosting enables code running on a platform model to directly access the I/O facilities on a host computer.

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.

### 3.10.2 Semihosting configuration

You can use different commands for the semihosting configuration.

The syntax to use on the command line to enable or disable semihosting is as follows:

```
./Foundation_Platform --(no-)semihost
```

The syntax to use on the command line to set the semihosting command-line string is as follows:

```
./Foundation_Platform --semihost-cmd=<command string>
```