

Fast Models

Version 11.4

Fixed Virtual Platforms (FVP) Reference Guide

The logo for Arm, consisting of the lowercase letters 'arm' in a bold, sans-serif font.

Fast Models

Fixed Virtual Platforms (FVP) Reference Guide

Copyright © 2014–2018 Arm Limited or its affiliates. All rights reserved.

Release Information

Document History

Issue	Date	Confidentiality	Change
A	31 May 2014	Non-Confidential	New document for Fast Models v9.0, from DUI0575H for v8.3.
B	30 November 2014	Non-Confidential	Update for v9.1.
C	28 February 2015	Non-Confidential	Update for v9.2.
D	31 May 2015	Non-Confidential	Update for v9.3.
E	31 August 2015	Non-Confidential	Update for v9.4.
F	30 November 2015	Non-Confidential	Update for v9.5.
G	29 February 2016	Non-Confidential	Update for v9.6.
H	31 May 2016	Non-Confidential	Update for v10.0.
I	31 August 2016	Non-Confidential	Update for v10.1.
J	11 November 2016	Non-Confidential	Update for v10.2.
K	17 February 2017	Non-Confidential	Update for v10.3.
1100-00	31 May 2017	Non-Confidential	Update for v11.0. Document numbering scheme has changed.
1101-00	31 August 2017	Non-Confidential	Update for v11.1.
1102-00	17 November 2017	Non-Confidential	Update for v11.2.
1103-00	23 February 2018	Non-Confidential	Update for v11.3.
1104-00	22 June 2018	Non-Confidential	Update for v11.4.

Non-Confidential 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, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

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

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is

not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

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

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) 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 <http://www.arm.com/company/policies/trademarks>.

Copyright © 2014–2018 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

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

Fast Models Fixed Virtual Platforms (FVP)

Reference Guide

	Preface	
	<i>About this book</i>	7
Chapter 1	Introduction	
	1.1 <i>About FVPs</i>	1-10
Chapter 2	Getting Started with Fixed Virtual Platforms	
	2.1 <i>Loading and running an application on an FVP</i>	2-12
	2.2 <i>Configuring the model</i>	2-13
	2.3 <i>FVP debug</i>	2-14
	2.4 <i>Using the VE CLCD window</i>	2-15
	2.5 <i>Ethernet with VE FVPs</i>	2-18
	2.6 <i>Using a terminal with a system model</i>	2-20
	2.7 <i>Virtio P9 device component</i>	2-22
Chapter 3	Programming Reference for Base FVPs	
	3.1 <i>Base - about</i>	3-24
	3.2 <i>Base Platform RevC changes</i>	3-25
	3.3 <i>BasePlatformPCIRevC component</i>	3-27
	3.4 <i>Base - memory</i>	3-29
	3.5 <i>Base - interrupt assignments</i>	3-33
	3.6 <i>Base - clocks</i>	3-36

3.7	Base - parameters	3-37
3.8	Base - components	3-38
3.9	Base - differences between the AEMv8-A FVP and core FVPs	3-46
3.10	Base - VE compatibility	3-47
3.11	Base - unsupported VE features	3-49

Chapter 4

Programming Reference for MPS2 FVPs

4.1	MPS2 - about	4-51
4.2	MPS2 platform types	4-52
4.3	MPS2 - memory maps	4-53
4.4	MPS2 - interrupt assignments	4-59
4.5	MPS2 - differences between models and hardware	4-60

Chapter 5

Programming Reference for VE FVPs

5.1	VE - about	5-62
5.2	Memory maps for VE FVPs	5-64
5.3	Interrupt maps for VE FVPs	5-68
5.4	VE parameters	5-70
5.5	VE - components	5-72
5.6	Differences between the VE hardware and the system model	5-78

Preface

This preface introduces the *Fast Models Fixed Virtual Platforms (FVP) Reference Guide*.

It contains the following:

- [About this book on page 7.](#)

About this book

Arm® Fixed Virtual Platform Reference. This manual introduces the Fixed Virtual Platforms, and describes how you can use them with other tools.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

This chapter introduces the document.

Chapter 2 Getting Started with Fixed Virtual Platforms

This chapter describes how to use FVPs.

Chapter 3 Programming Reference for Base FVPs

This chapter describes the memory map and the parameters for the peripheral and system component models.

Chapter 4 Programming Reference for MPS2 FVPs

This chapter describes the model of the hardware platform.

Chapter 5 Programming Reference for VE FVPs

This chapter describes the memory map and the parameters for the peripheral and system component models.

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 *Fast Models Fixed Virtual Platforms (FVP) Reference Guide*.
- The number 100966_1104_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

————— **Note** —————

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

Other information

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

Chapter 1

Introduction

This chapter introduces the document.

It contains the following section:

- [1.1 About FVPs on page 1-10.](#)

1.1 About FVPs

Fixed Virtual Platforms (FVPs) enable development of software without the requirement for real hardware.

FVPs are supplied as standalone executables for Linux and Windows. They are not customizable, although you can configure some aspects of their behavior through command-line parameters.

Arm provides different types of FVP. They can be based on the following:

- Armv8-A or Armv8-R Base Platform. These are called Base FVPs.
- Arm MPS2 or Arm MPS2+ platforms, for Cortex[®]-M series processors. These are called MPS2 FVPs.
- Arm Versatile™ Express development boards. These are called VE FVPs.

FVPs are available with a wide range of Armv7 and Armv8 processors, and support the CADI and MTI interfaces, so can be used for debugging and for trace output.

Another type of FVP is the Foundation Platform, which is a simple FVP that includes an Armv8-A AEM processor model, that is suitable for running bare-metal applications and for booting Linux.

Arm provides validated Linux and Android deliverables for the Armv8-A AEM Base Platform FVP and for the Foundation Platform. These are available on the Arm Community website at [Arm Development Platforms](#). To get started with Linux on Armv8-A FVPs, see [Armv8-A FVPs](#) also on Arm Community.

Chapter 2

Getting Started with Fixed Virtual Platforms

This chapter describes how to use FVPs.

It contains the following sections:

- *2.1 Loading and running an application on an FVP* on page 2-12.
- *2.2 Configuring the model* on page 2-13.
- *2.3 FVP debug* on page 2-14.
- *2.4 Using the VE CLCD window* on page 2-15.
- *2.5 Ethernet with VE FVPs* on page 2-18.
- *2.6 Using a terminal with a system model* on page 2-20.
- *2.7 Virtio P9 device component* on page 2-22.

2.1 Loading and running an application on an FVP

There are different ways to run an application on an FVP, for example from a command prompt, or from Model Debugger, or DS-5.

To run an FVP from the command prompt, change to the directory where your model is located. At the command prompt, enter the model name followed by the model options. To see all available options, use the `--help` option. The following options are commonly used when running an application on an FVP:

`-a filename.axf`

Specifies the application to load. The file can be in one of the following formats, or in gzip-compressed versions of them:

- ELF.
- Motorola S-Record.
- Intel-Hex.
- Verilog-Hex, in the format:

```
@<address_in_hex> <byte_in_hex>
```

`--data filename.bin@address`

Loads binary data into memory at the address you specify.

`-C instance.parameter=value`

Sets a single model parameter. Parameters are specified as a path that separates the instance names and the parameters using dots. For example, `-C bp.flashloader0.fname=fip.bin` loads a program into flash. To list all the available parameters, with their type, default value, and description, invoke the model with the `--list-params`, or `-l` option. To set multiple parameters at the same time, use the `-f` option instead.

`-f config_file.txt`

Specifies the name of a plain text configuration file. Configuration files simplify managing multiple model parameters. You can set the same parameters using this option as with the `-C` option.

`-S`

Starts a CADI debug server. This option allows a CADI-enabled debugger, such as Model Debugger or DS-5 Debugger, to connect to the running model. The model waits for the debugger to connect before starting.

For example:

```
models_directory/FVP_Base_Cortex-A57x1 -a __image.axf -f params.txt
```

You can also launch and debug bare metal and Linux applications on an FVP from Model Debugger or DS-5 Debugger. These debuggers use CADI to communicate with the FVP, so you must use the `-S` option when launching the FVP.

Starting the model opens the FVP CLCD display, which shows the contents of the simulated color LCD framebuffer.

Related information

[Arm DS-5 Debugger User Guide](#)

[Model Debugger for Fast Models User Guide](#)

2.2 Configuring the model

When you start the model from the command line, you can configure it using either:

- One or more `-C` command-line arguments.
- A configuration file and the `-f` command-line argument.

Each `-C` command-line argument or line in the configuration file must contain:

- The name of the component instance.
- The parameter to modify.
- Its value.

Use the following format:

instance.parameter=value

The `instance` can be a hierarchical path, with each level separated by a dot “.” character.

Note

- Comment lines in the configuration file begin with a `#` character.
 - You can set Boolean values using either `true` or `false`, or `1` or `0`.
-

You can generate a configuration file with all parameters set to default values by redirecting the output from the `--list-params` option into a new file, for example:

```
FVP_Base_AEMv8A.exe --list-params > params.txt
```

Example 2-1 Sample lines in a configuration file

```
# Disable semihosting using true/false syntax
coretile.cluster0.cpu0.semihosting-enable=false
#
# Enable VFP at reset using 1/0 syntax
coretile.cluster0.cpu0.vfp-enable_at_reset=1
#
# Set the baud rate for UART 0
motherboard.pl011_uart0.baud_rate=0x4800
```

2.3 FVP debug

This section describes how to debug an FVP.

FVP debug options

To debug an FVP, you can either:

- Run the FVP from within a CADI-enabled debugger.
- Start the FVP with the `-S` command-line argument and then connect a CADI-enabled debugger to it.

For information about using your debugger in these ways, see your debugger documentation.

Semihosting support

Semihosting enables code running on a platform model to directly access the I/O facilities on a host computer. Examples of these facilities include console I/O and file I/O.

The simulator handles semihosting by intercepting `HLT 0xF000`, `SVC 0x123456`, or `SVC 0xAB`, depending on whether the processor is in A64, A32 or T32 state. It handles all other HLTs and SVCs as normal.

If the operating system does not use `HLT 0xF000`, `SVC 0x123456`, or `SVC 0xAB` for its own purposes, it is not necessary to disable semihosting support to boot an operating system.

To temporarily or permanently disable semihosting support for a current debug connection, see your debugger documentation.

Related information

Arm Compiler Software Development Guide

2.4 Using the VE CLCD window

When an FVP starts, the FVP CLCD window opens, representing the contents of the simulated color LCD framebuffer. It automatically resizes to match the horizontal and vertical resolution that are set in the CLCD peripheral registers.



Figure 2-1 CLCD window in its default state at startup

The top section of the CLCD window displays the status information.

USERSW

Eight white boxes show the state of the VE User DIP switches:

These represent switch S6 on the VE hardware, USERSW[8:1], which is mapped to bits [7:0] of the SYS_SW register at address 0x1C010004.

The switches are in the off position by default. To change its state, click in the area above or below a white box.

BOOTSW

Eight white boxes show the state of the VE Boot DIP switches.

These represent switch S8 on the VE hardware, BOOTSEL[8:1], which is mapped to bits [15:8] of the SYS_SW register at address 0x1C010004.

The switches are in the off position by default.

————— **Note** —————

Arm recommends that you configure the Boot DIP switches using the `boot_switch` model parameter instead of using the CLCD interface. Changing Boot DIP switch positions while the model is running can result in unpredictable behavior.

S6LED

Eight colored boxes indicate the state of the VE User LEDs.

These represent LEDs D[21:14] on the VE hardware, which are mapped to bits [7:0] of the SYS_LED register at address 0x1C010008. The boxes correspond to the red/yellow/green LEDs on the VE hardware.

Total Instr

A counter showing the total number of instructions executed.

Because the FVP models provide a *Programmer's View* (PV) of the system, the CLCD displays total instructions rather than total processor cycles. Timing might differ substantially from the hardware because:

- Bus fabric is simplified.
- Memory latencies are minimized.
- Cycle approximate processor and peripheral models are used.

In general, bus transaction timing is consistent with the hardware, but the timing of operations within the model is not accurate.

Total Time

A counter showing the total elapsed time, in seconds.

This time is wall clock time, not simulated time.

Rate Limit

A feature that disables or enables fast simulation.

Because the system model is highly optimized, your code might run faster than it would on real hardware. This effect might cause timing issues.

Rate Limit is enabled by default. Simulation time is restricted so that it more closely matches real time.

To disable or enable Rate Limit, click the square button. When you disable Rate Limit, the text changes from ON to OFF and the colored box becomes darker. You can configure this option when instantiating the model with the `rate_limit-enable` visualization component parameter.

When you click the **Total Instr** or **Total Time** items in the CLCD, the display changes to show **Instr/sec** (instructions per second) and **Perf Index** (performance index).

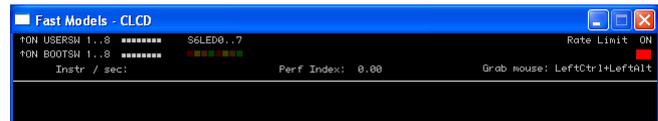


Figure 2-2 CLCD window with Rate Limit ON, showing Instr/sec and Perf Index

You can click the items again to toggle between the original and alternative displays.

Instr/sec

The number of instructions that execute 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.

You can reset the simulation counters by resetting the model.

The VE FVP CLCD displays the core run state for each core with a colored icon. The icons are to the left of the **Total Instr** (or **Inst/sec**) item. They appear when you start the simulation.



Figure 2-3 Core run state icons for a quad core model

Table 2-1 Core run state icon descriptions

Icon	State label	Description
?	UNKNOWN	Run status unknown, that is, simulation has not started.
▶	RUNNING	Core running, is not idle, and is executing instructions.
⏸	HALTED	External halt signal asserted.
E	STANDBY_WFE	Last instruction executed was WFE and standby mode has been entered.
I	STANDBY_WFI	Last instruction executed was WFI and standby mode has been entered.

Table 2-1 Core run state icon descriptions (continued)

Icon	State label	Description
	IN_RESET	External reset signal asserted.
	DORMANT	Partial core power down.
	SHUTDOWN	Complete core power down.

If the CLCD window has focus:

- Any keyboard input is translated to PS/2 keyboard data.
- Any mouse activity over the window is translated into PS/2 relative mouse motion data. The data is then streamed to the KMI peripheral model FIFOs.

————— **Note** —————

The simulator only sends relative mouse motion events to the model. As a result, the host mouse pointer does not necessarily align with the target OS mouse pointer.

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 **right Ctrl** key does not have the same effect.

If you prefer to use a different key, configure it with the `trap_key` visualization component parameter.

Related reference

VEVisualisation - parameters on page 5-74

2.5 Ethernet with VE FVPs

This section describes how to use Ethernet with VE FVPs.

Using Ethernet with VE FVPs

The VE FVPs have a virtual Ethernet component. This component is a model of the SMSC 91C111 Ethernet controller, and uses a TAP device to communicate with the network. By default, the Ethernet component is disabled.

Host requirements

Before you can use the Ethernet capability of VE FVPs, set up your host computer.

Target requirements

This section describes the target requirements.

Target requirements - about

The VE FVPs include a software implementation of the SMSC 91C111 Ethernet controller. Your target OS must therefore include a driver for this specific device. To use the SMSC chip, configure the kernel. Linux supports the SMSC 91C111.

The configurable SMSC 91C111 component parameters are:

- `enabled`.
- `mac_address`.
- `promiscuous`.

`enabled`

When the device is disabled, the kernel cannot detect the device.

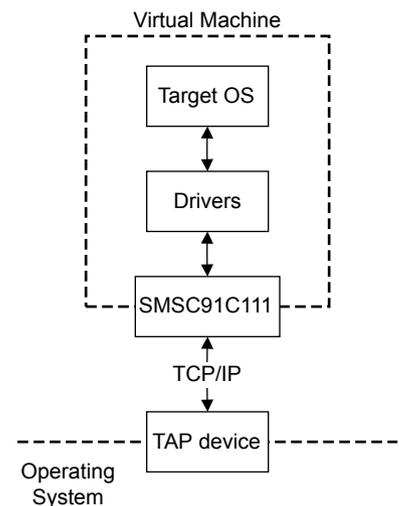


Figure 2-4 Model networking structure block diagram

To perform read and write operations on the TAP device, configure a HostBridge component. The HostBridge component is a virtual *Programmer's View* (PV) model. It acts as a networking gateway to exchange Ethernet packets with the TAP device on the host, and to forward packets to NIC models.

mac_address

There are two options for the `mac_address` parameter.

If a MAC address is not specified, when the simulator is run it takes the default MAC address, which is randomly generated. This random generation provides some degree of MAC address uniqueness when running models on multiple hosts on a local network.

promiscuous

The Ethernet component starts in promiscuous mode by default. In this mode, it receives all network traffic, even any not addressed to the device. Use this mode if you are using a single network device for multiple MAC addresses. Use this mode if, for example, you share the network card between your host OS and the VE FVP Ethernet component.

By default, the Ethernet device on the VE FVP has a randomly generated MAC address and starts in promiscuous mode.

2.6 Using a terminal with a system model

The Terminal component is a virtual component that enables UART data to be transferred between a TCP/IP socket on the host and a serial port on the target.

————— **Note** —————

To use the Terminal component with a Microsoft Windows 7 client, you must first install Telnet. The Telnet application is not installed on Microsoft Windows 7 by default.

Download the application by following the instructions on the Microsoft web site. Search for “Windows 7 Telnet” to find the Telnet FAQ page. To install Telnet:

1. Select **Start > Control Panel > Programs and Features** to open a window that enables you to uninstall or change programs.
2. Select **Turn Windows features on or off** on the left side of the bar. This opens the Microsoft Windows Features dialog. Select the **Telnet Client** check box.
3. Click **OK**. The installation of Telnet might take several minutes to complete.

The following figure shows a block diagram of one possible relationship between the target and host through the Terminal component. The TelnetTerminal block is what you configure when you define Terminal component parameters. The Virtual Machine is your FVP.

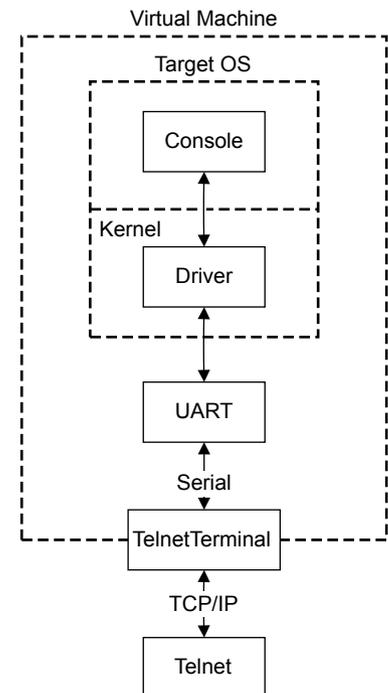


Figure 2-5 Terminal block diagram

On the target side, the console process that is invoked by your target OS relies on a suitable driver being present. Such drivers are normally part of the OS kernel. The driver passes serial data through a UART. The data is forwarded to the TelnetTerminal component, which exposes a TCP/IP port to the world outside of the FVP. This port can be connected to by, for example, a Telnet process on the host.

By default, the FVP starts four telnet Terminals when the model is initialized. You can change the startup behavior for each of the four Terminals by modifying the corresponding component parameters.

If the Terminal connection is broken, for example by closing a client telnet session, the port is re-opened on the host. This might have a different port number if the original one is no longer available. Before the first data access, you can connect a client of your choice to the network socket. If there is no existing

connection when the first data access is made, and the `start_telnet` parameter is `true`, a host telnet session is started automatically.

The port number of a particular Terminal instance can be defined when the FVP starts. The actual value of the port that is used by each Terminal is declared when it starts or restarts, and might not be the value that you specified if the port is already in use. If you are using Model Shell, the port numbers are displayed in the host window in which you started the model.

You can start the Terminal component in either telnet mode or raw mode.

Telnet mode

In telnet mode, the Terminal component supports a subset of the RFC 854 protocol. This means that the Terminal participates in negotiations between the host and client concerning what is and is not supported, but flow control is not implemented.

Raw mode

Raw mode enables the byte stream to pass unmodified between the host and the target. This means that the Terminal component does not participate in initial capability negotiations between the host and client. It acts as a TCP/IP port. You can use this feature to directly connect to your target through the Terminal component.

2.7 Virtio P9 device component

The VirtioP9Device component is included in Base, BaseR, and A-profile VE platforms. It implements a subset of the Plan 9 file protocol over a virtio transport. It enables accessing a directory on the host's filesystem within Linux, or another operating system that implements the protocol, running on a platform model.

Setting up VirtioP9Device

Take the following steps to use this component:

- Use a version of Linux that supports v9fs over virtio and virtio-mmio devices.
- Update the device tree to include the VirtioP9Device component, or specify it on the kernel command-line, as shown below. The address range for both VE and Base platforms is 0x1C140000-0x1C14FFFF.

The interrupt number is 43, or IRQ 75, for both VE and Base platforms.

- Set the following parameter to the directory on the host that you want to mount in the model:

VE:

```
motherboard.virtiop9device.root_path
```

Base:

```
bp.virtiop9device.root_path
```

- On Linux, mount the host directory by using the following command in the model:

```
$ mount -t 9p -o trans=virtio,version=9p2000.L FM <mount point>
```

Example kernel command-line argument

```
virtio_mmio.device=0x10000@0x1c140000:75
```

Example entry for DTS files

Add this entry next to the corresponding virtio_block entry:

```
virtio_p9@0140000 {  
    compatible = "virtio,mmio";  
    reg = <0x0 0x1c140000 0x0 0x1000>;  
    interrupts = <0x0 0x2b 0x4>;  
};
```

Chapter 3

Programming Reference for Base FVPs

This chapter describes the memory map and the parameters for the peripheral and system component models.

It contains the following sections:

- [3.1 Base - about](#) on page 3-24.
- [3.2 Base Platform RevC changes](#) on page 3-25.
- [3.3 BasePlatformPCIRevC component](#) on page 3-27.
- [3.4 Base - memory](#) on page 3-29.
- [3.5 Base - interrupt assignments](#) on page 3-33.
- [3.6 Base - clocks](#) on page 3-36.
- [3.7 Base - parameters](#) on page 3-37.
- [3.8 Base - components](#) on page 3-38.
- [3.9 Base - differences between the AEMv8-A FVP and core FVPs](#) on page 3-46.
- [3.10 Base - VE compatibility](#) on page 3-47.
- [3.11 Base - unsupported VE features](#) on page 3-49.

3.1 Base - about

The Base Platform system model allows early development, distribution, and demonstration of software deliverables. A range of Base FVPs are supplied as standalone products and as examples in Fast Models.

The standard peripheral set enables software development and porting. The platform is an evolution of the VE *Fixed Virtual Platforms* (FVPs), based on the Arm Versatile Express (VE) hardware development platform.

The Base Platform system model provides:

- Two configurable clusters of up to four core models that implement:
 - AArch64 at all exception levels.
 - Configurable AArch32 support at all exception levels.
 - Configurable support for little and big endian at all exception levels.
 - Generic timers.
 - Self-hosted debug.
 - CADI debug.
 - GICv3 memory-mapped processor interfaces and distributor.
- Peripherals for multimedia or networking environments.
- Four PL011 UARTs.
- A CoreLink™ CCI-400 Cache Coherent Interconnect.
- Architectural GICv3 model.
- High Definition LCD Display Controller, 1920×1080 resolution at 60fps, with single I2S and four stereo channels.
- 64MB NOR flash and board peripherals.
- CoreLink TZC-400 TrustZone® Address Space Controller.

To run FVP_Base_AEMv8A-AEMv8A-MMU500+DMA330* models, reconfigure the DMAC or SMMU ranges because the default ranges for these components overlap.

3.2 Base Platform RevC changes

The Base Platform RevC includes the following changes from the previous revision.

- It includes a PCIe config region and two PCIe memory regions. See the memory map for details.
- It includes a CoreLink CCI-550 Cache Coherent Interconnect.
- It includes an SMMUv3 AEM. This is an architectural model that implements the SMMUv3.0 and SMMUv3.1 architectures. These architectures are for I/O virtualization of devices. The SMMU is placed so that accesses to memory by PCI devices acting as bus masters are affected by it.

The SMMU has the following features:

- Memory that is mapped to the range `0x2B400000-0x2B4FFFFF`.
- Interrupts with IRQ IDs in the range 103-111.
- The event output pin of the SMMU is passed to the clusters.
- The downstream ports of the SMMU attach to the coherent bus infrastructure and so are coherent with the core clusters. All cores and the SMMU are in the same shareability domain. There is no distinction between the inner and outer shareability domains.
- The parameters of the SMMU determine its capabilities and have default values which can be overwritten if necessary.
- The SMMU is configured to only accept 16-bit StreamIDs and there is a 1:1 correspondence between RequestorID and StreamID.
- By default, the SMMU uses DeviceID `0x10000` to identify itself to the GIC (`pci.pci_smmuv3.mmu.smmu_msi_device_id`).

The parameter `gic_distributor.ITS-device-bits` is set to 17 by default to support the 17-bit DeviceIDs.

The SMMU has the following limitations:

- It does not support RAS.
- The PMU has limited functionality. Only a subset of the architecturally mandatory events are supported, as indicated by the `SMMU_PMCG_CEID0` fields. The PMU is intended for demonstration purposes only and for driver development.
- Two PCIe virtio devices are above the SMMU. By default they are configured to be device 0 and 2 on bus 0.
- The PCIe devices use a DeviceID that is the same as their RequestorID (BDF).
- Legacy PCI interrupts:
 - Each PCI device is hardwired to use INTA, with a value of 1 in the `interrupt_pin` register. This is required by the PCI specification for single-function devices.
 - The interrupts in the PCI host bridge are mapped according to section 2.2.6 of the PCI Local Bus Specification Revision 3.0, using the following formula, where the values for DeviceInterrupt are INTA = 0, INTB = 1, INTC = 2, INTD = 3:

$$\text{BridgeInterrupt} = (\text{Device} + \text{DeviceInterrupt}) \% 4$$

This formula produces the following mappings:

BridgeInterrupt ID	DeviceInterrupt
200	INTA
201	INTB
202	INTC
203	INTD
207	SERR

- The model optionally implements MSI-X, depending on whether a parameter is set. If this parameter is set, an MSI-X capability is advertised as a PCI capability.

————— **Note** —————

The virtio specification is not fully compliant with the PCI specification and the virtio block device cannot be used in a pure-polling mode where MSI-X is always masked and only polling the pending bit array is used.

The MSIs produced by the models, when directed to the GIC, have their payload rewritten to carry the DeviceID of the originating device to the GIC.

- The processor models implement architecture version v8.0, which does not support the Statistical Profiling Extension (SPE). To include SPE, add parameter `cluster0.has_arm_v8-3=1`, or similar, to the command line.

Related reference

[3.4.2 Base - memory map on page 3-29](#)

[3.5 Base - interrupt assignments on page 3-33](#)

3.3 BasePlatformPCIRevC component

This component is an integrated PCIe subsystem which forms part of the Base Platform RevC. It incorporates an SMMUv3, a PCIe, and two PCI devices which wrap a pair of virtio PCI block devices. This model is written in LISA+.

About BasePlatformPCIRevC

Note

- You can include this component in a platform model, but Arm does not support using its subcomponents to create your own PCIe platform.
- The PCIe is not an implementation of any specific IP, but a functional, and limited, implementation of the PCIe standard.

BasePlatformPCIRevC is composed of the following model components:

pci.pvbus2pci

The bridge from the Programmer's View bus to the PCI bus.

pci.pcidevice<n>

A wrapper around the underlying virtio block device. There are two block devices in the system, 0 and 1.

pci.pcivirtioblockdevice<n>

The instances of the virtio block device component.

Some interesting options are:

- If the following options are set to non-zero values, they print messages about the operation of the bridge. The higher the value, the more verbose the component is:

```
pci.pvbus2pci.diagnostics=0x0 # (int) default = '0x0': Diagnostics level: [0x0..0x4]
pci.pcidevice<N>.diagnostics=0x0 # (int) default = '0x0': Diagnostics level: [0x0..0x4]
```

- Each PCI device uses three BARs; one for config space, one for the MSI-X table structure and one for the MSI-X Pending Bit Array. Each of these can be configured to be 32 bits or 64 bits wide.

The Bus and Device number can be configured for each PCI device. If the device advertises MSI-X, support can be configured.

```
pci.pcidevice<N>.bus=0x0 # (int) default = '0x0' : Bus number for this device : [0x0..0xFF]
pci.pcidevice<N>.device=0x0 # (int) default = '0x0' : Device number on this bus : [0x0..0x1F]
pci.pcidevice<N>.bar0_64bit=0 # (bool) default = '0' : If BAR 0 is 64 bits wide, if region size is nonzero
pci.pcidevice<N>.msix_support=0 # (bool) default = '0' : Enable device support for MSI-X
pci.pcidevice<N>.bar2_64bit=0 # (bool) default = '0' : If BAR 2 is 64 bits wide, if region size is nonzero
pci.pcidevice<N>.bar4_64bit=0 # (bool) default = '0' : If BAR 4 is 64 bits wide, if region size is nonzero
```

- The following option configures the image file that the virtio block device exposes:

```
pci.pcivirtioblockdevice<N>.image_path="" # (string) default = '' : image file path
```

- There are two PVBUSLoggers in the pvbus2pci component. One is in front of the Configuration space and one is in front of the Device space:

```
FVP_Base_AEMv8A_AEMv8A-PCI.pci.pvbus2pci.devicelogger
FVP_Base_AEMv8A_AEMv8A-PCI.pci.pvbus2pci.cfglogger
```

- There is one PVBUSLogger in the pcidevice component. This reports on DMA accesses by the PCI device:

```
FVP_Base_AEMv8A_AEMv8A-PCI.pci.pcidevice0.dmalogger
```

- There is a PVBUSLogger downstream of the SMMU. This reports on the transactions after they have been transformed by the SMMU:

```
FVP_Base_AEMv8A_AEMv8A-PCI.pci.smmulogger
```

For example, you can see all accesses to device space by adding the following options to the command line:

```
--plugin GenericTrace.so
-C TRACE.GenericTrace.trace-sources="FVP_Base_AEMv8A_AEMv8A-PCI.pci.pvbus2pci.devicelogger.*"
```

Table 3-1 BasePlatformPCIRevC ports

Name	Protocol	Type	Description
pvbus_address_map_s	PVBus	Slave	Input port to service transactions based on the PVBus protocol.
pvbus_address_map_m	PVBus	Master	Output port to send out PVBus transactions that are not handled by this component.
system_reset	Signal	Slave	Input port to handle reset signals. It is used to reset the internal state of this component.
sev_out	Signal	Peer	Port to send out a notification of the occurrence of an event as sg::Signal to a peer.
interrupts[224]	Signal	Master	Array of output ports of type sg::Signal to send out interrupts generated by this component.
pvbus_pci_dma_m	PVBus	Master	Output port to send out any DMA (of PVBus protocol) accesses originating from this component.
clk_in	ClockSignal	Slave	Input port to connect to a ClockSignal provider.

Table 3-2 BasePlatformPCIRevC parameters

Name	Type	Allowed values	Default value	Description
ITS0-base	uint64_t	0-0xFFFFFFFFFFFFFFFF	0x2f020000	The ITS0 Base address.
pci_smmuv3.mmu.SMMU_IDR1	uint32_t	0-0xFFFFFFFF	0xe739d10	SMMU_IDR1.
pci_smmuv3.mmu.smmu_msi_device_id	uint32_t	0-0xFFFFFFFF	0x10000	When appropriately enabled, assume that MSIs that are generated by the SMMU are presented to the GIC with this DeviceID. See parameter <code>msi_attribute_transform</code> and <code>enable_device_id_checks</code> .

Related information

SMMUv3AEM component

VirtioBlockDevice component

3.4 Base - memory

This section describes the memory of the Base Platform.

This section contains the following subsections:

- [3.4.1 Base - secure memory on page 3-29.](#)
- [3.4.2 Base - memory map on page 3-29.](#)
- [3.4.3 Base - DRAM on page 3-32.](#)

3.4.1 Base - secure memory

Enable security checking on memory transactions by the TZC-400 by using the `bp.secure_memory` parameter.

Table 3-3 Secure and Non-secure access permissions

Security	<code>bp.secure_memory=false</code>	<code>bp.secure_memory=true</code>
S	Secure and Non-secure access permitted.	Secure access is permitted, Non-secure access aborts.
S/NS	Secure and Non-secure access permitted.	Secure and Non-secure access permitted.
P	Secure and Non-secure access permitted.	Access conditions are programmable by the TZC-400.

————— **Note** —————

The default state of the TZC-400 is to abort all accesses, even from Secure state.

Table 3-4 NSAIDs and filters that masters present to the TZC-400

Component	NSAID ^a	Filter
Cluster 0	9	0
Cluster 1	9	0
VirtIO	8	0
HDLCD0	2	2
CLCD	1	2

3.4.2 Base - memory map

The basis of this map is the Versatile Express RS2 memory map with extensions.

Table 3-5 Base Platform memory map

Peripheral	Start address	Size	End address	Security
Trusted Boot ROM, secure flash, IntelStrataFlashJ3	0x00_0000_0000	64MB	0x00_03FF_FFFF	S
Trusted SRAM	0x00_0400_0000	256KB	0x00_0403_FFFF	S
Trusted DRAM	0x00_0600_0000	32MB	0x00_07FF_FFFF	S
NOR flash, flash0, IntelStrataFlashJ3	0x00_0800_0000	64MB	0x00_0BFF_FFFF	S/NS
NOR flash, flash1, IntelStrataFlashJ3	0x00_0C00_0000	64MB	0x00_0FFF_FFFF	S/NS
PSRAM ^b	0x00_1400_0000	64MB	0x00_17FF_FFFF	S/NS

^a Non-Secure Access IDentity.

Table 3-5 Base Platform memory map (continued)

Peripheral	Start address	Size	End address	Security
VRAM	0x00_1800_0000	32MB	0x00_19FF_FFFF	S/NS
Ethernet, SMSC 91C111	0x00_1A00_0000	16MB	0x00_1AFF_FFFF	S/NS
USB, unimplemented	0x00_1B00_0000	16MB	0x00_1BFF_FFFF	S/NS
VE System Registers	0x00_1C01_0000	64KB	0x00_1C01_FFFF	S/NS
System Controller, SP810	0x00_1C02_0000	64KB	0x00_1C02_FFFF	S/NS
AACI, PL041	0x00_1C04_0000	64KB	0x00_1C04_FFFF	S/NS
MCI, PL180	0x00_1C05_0000	64KB	0x00_1C05_FFFF	S/NS
KMI - Keyboard, PL050	0x00_1C06_0000	64KB	0x00_1C06_FFFF	S/NS
KMI - Mouse, PL050	0x00_1C07_0000	64KB	0x00_1C07_FFFF	S/NS
UART0, PL011	0x00_1C09_0000	64KB	0x00_1C09_FFFF	S/NS
UART1, PL011	0x00_1C0A_0000	64KB	0x00_1C0A_FFFF	S/NS
UART2, PL011	0x00_1C0B_0000	64KB	0x00_1C0B_FFFF	S/NS
UART3, PL011	0x00_1C0C_0000	64KB	0x00_1C0C_FFFF	S/NS
Watchdog, SP805	0x00_1C0F_0000	64KB	0x00_1C0F_FFFF	S/NS
Base Platform Power Controller	0x00_1C10_0000	64KB	0x00_1C10_FFFF	S/NS
Dual-Timer 0, SP804	0x00_1C11_0000	64KB	0x00_1C11_FFFF	S/NS
Dual-Timer 1, SP804	0x00_1C12_0000	64KB	0x00_1C12_FFFF	S/NS
Virtio block device	0x00_1C13_0000	64KB	0x00_1C13_FFFF	S/NS
Virtio Plan 9 device	0x00_1C14_0000	64KB	0x00_1C14_FFFF	S/NS
Virtio net device	0x00_1C15_0000	64KB	0x00_1C15_FFFF	S/NS
Real-time Clock, PL031	0x00_1C17_0000	64KB	0x00_1C17_FFFF	S/NS
CF Card, unimplemented	0x00_1C1A_0000	64KB	0x00_1C1A_FFFF	S/NS
Color LCD Controller, PL111	0x00_1C1F_0000	64KB	0x00_1C1F_FFFF	S/NS
Non-trusted ROM, nontrustedrom	0x00_1F00_0000	4KB	0x00_1F00_0FFF	S/NS
CoreSight™ and peripherals	0x00_2000_0000	128MB	0x00_27FF_FFFF	S/NS
REFCLK CNTControl, Generic Timer	0x00_2A43_0000	64KB	0x00_2A43_FFFF	S
EL2 Generic Watchdog Control	0x00_2A44_0000	64KB	0x00_2A44_FFFF	S/NS
EL2 Generic Watchdog Refresh	0x00_2A45_0000	64KB	0x00_2A45_FFFF	S/NS
Trusted Watchdog, SP805	0x00_2A49_0000	64KB	0x00_2A49_FFFF	S
TrustZone Address Space Controller, TZC-400	0x00_2A4A_0000	64KB	0x00_2A4A_FFFF	S
REFCLK CNTRead, Generic Timer	0x00_2A80_0000	4KB	0x00_2A80_0FFF	S/NS
AP_REFCLK CNTCTL, Generic Timer	0x00_2A81_0000	4KB	0x00_2A81_0FFF	S/NS
AP_REFCLK CNTBase0, Generic Timer	0x00_2A82_0000	4KB	0x00_2A82_0FFF	S
AP_REFCLK CNTBase1, Generic Timer	0x00_2A83_0000	4KB	0x00_2A83_0FFF	S/NS

^b The device is implemented as RAM and is 8MB in size.

Table 3-5 Base Platform memory map (continued)

Peripheral	Start address	Size	End address	Security
DMC-400 CFG, unimplemented	0x00_2B0A_0000	64KB	0x00_2B0A_FFFF	S/NS
SMMUv3 AEM ^c	0x00_2B40_0000	1MB	0x00_2B4F_FFFF	S/NS
GIC Physical CPU interface, GICC ^d	0x00_2C00_0000	8KB	0x00_2C00_1FFF	S/NS
GIC Virtual Interface Control, GICH ^d	0x00_2C01_0000	4KB	0x00_2C01_0FFF	S/NS
GIC Virtual CPU Interface, GICV ^d	0x00_2C02_F000	8KB	0x00_2C03_0FFF	S/NS
CCI-400	0x00_2C09_0000	64KB	0x00_2C09_FFFF	S/NS
Non-trusted SRAM	0x00_2E00_0000	64KB	0x00_2E00_FFFF	S/NS
GICv3 IRI GICD ^d	0x00_2F00_0000	64KB	0x00_2F00_FFFF	S/NS
GICv3 IRI GITS ^d	0x00_2F02_0000	128KB	0x00_2F03_FFFF	S/NS
GICv3 IRI GICR ^d	0x00_2F10_0000	1MB	0x00_2F1F_FFFF	S/NS
PCIe config region ^c	0x00_4000_0000	256MB	0x00_4FFF_FFFF	S/NS
PCIe memory region 1 ^c	0x00_5000_0000	256MB	0x00_5FFF_FFFF	S/NS
Trusted Random Number Generator	0x00_7FE6_0000	4KB	0x00_7FE6_0FFF	S
Trusted Non-volatile counters	0x00_7FE7_0000	4KB	0x00_7FE7_0FFF	S
Trusted Root-Key Storage	0x00_7FE8_0000	4KB	0x00_7FE8_0FFF	S
DDR3 PHY, unimplemented	0x00_7FEF_0000	64KB	0x00_7FEF_FFFF	S/NS
HD LCD Controller, PL370	0x00_7FF6_0000	64KB	0x00_7FF6_FFFF	S/NS
DRAM, 0GB-2GB	0x00_8000_0000	2GB	0x00_FFFF_FFFF	P
DRAM, 2GB-32GB	0x08_8000_0000	30GB	0x0F_FFFF_FFFF	P
PCIe memory region 2 ^c	0x40_0000_0000	256GB	0x7F_FFFF_FFFF	S/NS
DRAM, 32GB-512GB	0x88_0000_0000	480GB	0xFF_FFFF_FFFF	P
DRAM, 512GB-8TB	0x00_0880_0000_0000	7TB	0x00_0FFF_FFFF_FFFF	P
DRAM, 8TB-128TB	0x00_8800_0000_0000	120TB	0x00_FFFF_FFFF_FFFF	P
DRAM, 128TB-2PB	0x08_8000_0000_0000	1920TB	0x0F_FFFF_FFFF_FFFF	P
DRAM, 2PB-4PB	0x88_0000_0000_0000	2PB	0x8F_FFFF_FFFF_FFFF	P

Note

The BaseR platform copies its memory map from the Base platform, but swaps the upper 2GB of address space with the lower 2GB. Therefore any peripherals in the memory range [0x0-0x7FFFFFFF] in Base are available at the same offset in the memory range [0x80000000-0xFFFFFFFF] in BaseR. Any peripherals in the memory range [0x80000000-0xFFFFFFFF] in Base are available at the same offset in the memory range [0x0-0x7FFFFFFF] in BaseR. The DRAM in the Base platform memory map starts at address 0x80000000, which in BaseR would prevent any code from running from DRAM after reset. The code would be prevented from running because in the Armv8-R architecture the upper 2GB of memory does not have execution permissions by default.

^c Base Platform RevC only

^d You can configure the address of this region using parameters to the model. See the parameters in section *GICv3IRI component of Fast Models components* in the Fast Models Reference Manual.

3.4.3 Base - DRAM

The multiple DRAM regions do not alias each other and form a contiguous 4PB area. The total amount of DRAM on the Base Platform system model is configurable. This ability affects where usable DRAM appears.

If the Base Platform system model has `bp.dram_size=4`, the default, then 2GB of DRAM is accessible at `0x00_8000_0000` to `0x00_FFFF_FFFF`, and the remaining 2GB is accessible at `0x08_8000_0000` to `0x08_FFFF_FFFF`.

If, instead, the Base Platform system model has `bp.dram_size=8`, then 2GB of DRAM is accessible at `0x00_8000_0000` to `0x00_FFFF_FFFF` and the remaining 6GB is accessible at `0x08_8000_0000` to `0x09_FFFF_FFFF`.

The default contents of RAM not otherwise written by the simulation is a repeating sequence of the following 64-bit value: `0xCFDFDFDFDFDFCF`.

————— **Note** —————

Memory is allocated on demand, and performance degrades if very large amounts of memory are used.

—————

3.5 Base - interrupt assignments

The platform assigns the *Shared Peripheral Interrupts* (SPIs) and *Private Peripheral Interrupts* (PPIs) on the GIC.

————— **Note** —————

- SPI and PPI numbers are mapped onto GIC interrupt IDs as the *Arm® Generic Interrupt Controller Specification* describes.
- IRQ IDs 103-111 and 200-207 apply to the Base Platform RevC only.

Table 3-6 SPI GIC assignments

IRQ ID	SPI offset	Device
32	0	Watchdog, SP805.
34	2	Dual-Timer 0, SP804.
35	3	Dual-Timer 1, SP804.
36	4	Real-time Clock, PL031.
37	5	UART0, PL011.
38	6	UART1, PL011.
39	7	UART2, PL011.
40	8	UART3, PL011.
41	9	MCI, PL180, MCIINTR0.
42	10	MCI, PL180, MCIINTR1.
43	11	AACI, PL041.
44	12	KMI - Keyboard, PL050.
45	13	KMI - Mouse, PL050.
46	14	Color LCD Controller, PL111.
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 P9 device.
76	44	Virtio net device.
80	48	TZC-400.
92	60	cluster0.cpu0 PMUIRQ.
93	61	cluster0.cpu1 PMUIRQ.
94	62	cluster0.cpu2 PMUIRQ.

Table 3-6 SPI GIC assignments (continued)

IRQ ID	SPI offset	Device
95	63	cluster0.cpu3 PMUIRQ.
96	64	cluster1.cpu0 PMUIRQ.
97	65	cluster1.cpu1 PMUIRQ.
98	66	cluster1.cpu2 PMUIRQ.
99	67	cluster1.cpu3 PMUIRQ.
103 ^e	71	SMMUv3 non-secure combined interrupt.
104 ^e	72	SMMUv3 secure combined interrupt. Unused because there is no secure side.
105 ^e	73	SMMUv3 secure event queue. Unused because there is no secure side.
106 ^e	74	SMMUv3 non-secure event queue.
107 ^e	75	SMMUv3 PRI queue. Unused because no PCIe device supports PRI.
108 ^e	76	SMMUv3 secure command queue sync. Unused because there is no secure side.
109 ^e	77	SMMUv3 non-secure command queue sync.
110 ^e	78	SMMUv3 secure GERROR. Unused because there is no secure side.
111 ^e	79	SMMUv3 non-secure GERROR.
117	85	HD LCD Controller, PL370.
139	107	Trusted Random Number Generator.
200 ^e	168	PCIe INTA.
201 ^e	169	PCIe INTB.
202 ^e	170	PCIe INTC.
203 ^e	171	PCIe INTD.
207 ^e	175	PCIe SERR.

Table 3-7 PPI GIC assignments

IRQ ID	PPI offset	Device
19	3	Secure hypervisor virtual timer interrupt
20	4	Secure hypervisor physical timer interrupt
22	6	DCC, comms channel, interrupt
23	7	PMU, performance counter, overflow
24	8	CTI, Cross Trigger Interface, interrupt
25	9	Virtual CPU interface maintenance interrupt
26	10	Hypervisor timer interrupt
27	11	Virtual timer interrupt
28	12	Hypervisor virtual timer interrupt

^e Base Platform RevC only

Table 3-7 PPI GIC assignments (continued)

IRQ ID	PPI offset	Device
29	13	Secure physical timer interrupt
30	14	Non-secure physical timer interrupt

3.6 Base - clocks

This section describes the clock frequencies of the Base Platform peripherals.

Table 3-8 Peripheral clock frequencies in the Base Platform

Device	Clock
Clusters	100MHz
REFCLK CNTControl, Generic Timer	100MHz
AP_REFCLK CNTCTL, Generic Timer	100MHz
Dual-Timer 0-1, SP804	35MHz
VE system registers	24MHz
UART 0-3, PL011	24MHz
KMI 0-1, PL050	24MHz
MCI, PL180	24MHz
AACI, PL041	24MHz
Ethernet, SMSC 91C111	24MHz
Watchdog, SP805	24MHz
Color LCD Controller, PL111	23.75MHz
HD LCD Controller, PL370	10MHz
Trusted Watchdog, SP805	32.768kHz
Real-time Clock, PL031	1Hz

3.7 Base - parameters

This section describes the parameters.

Table 3-9 Base Platform parameters

Parameter	Type	Allowed values	Default value	Description
bp.dram_size	int	2, 4, or 8-4000000	4	Size of main memory in gigabytes: 2, 4, or any value between 8 and 4000000.
bp.proc_idn ^f	uint32_t	-	-. ^g	Processor ID for VE_SysRegs SYS_PROCIDn.
bp.secure_memory	bool	true, false	true	The security state of the processor limits access to peripherals and RAM.
bp.variant ^f	uint32_t	0x0-0xF	-. ^g	Board variant for VE_SysRegs SYS_ID.
cache_state_modelled	bool	true, false	true	Enable d-cache and i-cache state for all components.

Table 3-10 Base Platform debug parameters

Parameter	Type	Allowed values	Default value	Description
dbgen	bool	true, false	true	Debug authentication signal, dbgen .
niden	bool	true, false	true	Debug authentication signal, niden .
spiden	bool	true, false	true	Debug authentication signal, spiden .
spniden	bool	true, false	true	Debug authentication signal, spniden .

^f Some platforms do not expose this parameter.

^g Platform specific.

3.8 Base - components

This section describes the components.

This section contains the following subsections:

- [3.8.1 Base - components - about](#) on page 3-38.
- [3.8.2 Base - Base_PowerController component](#) on page 3-38.
- [3.8.3 Base - DebugAccessPort component](#) on page 3-42.
- [3.8.4 Base - simulator visualization component](#) on page 3-43.
- [3.8.5 Base - VE_SysRegs component](#) on page 3-44.

3.8.1 Base - components - about

These component models implement some of the functionality of the Versatile Express (VE) hardware.

A complete model implementation of a Base Platform system model includes both Base Platform-specific components and generic components such as buses and timers.

3.8.2 Base - Base_PowerController component

This section describes the Base_PowerController component.

Base_PowerController - control interface

The Base_PowerController provides a basic register interface for software to control the power-up and power-down of cores in the cluster.

Identify cores in the system to the Base_PowerController by writing 24 bits in MPIDR format, providing the following levels of affinity:

Bits [23:16]

Affinity level 2.

Bits [15:8]

Affinity level 1.

Bits [7:0]

Affinity level 0.

Examples of affinity usage are `not_applicable/cluster/processor` and `cluster/processor/thread`.

Base_PowerController - parameters

This section describes the parameters.

Table 3-11 Base_PowerController parameters

Parameter	Allowed values	Default value	Description
startup	-	'0.0.0.*'	A comma-separated list of cores to power up at startup or system reset. Specify core affinities with a dotted-quad ('0.0.0.0' refers to cluster0.cpu0 and '0.0.1.1' refers to cluster1.cpu1). Use wildcards to indicate all cores at an affinity level ('0.0.0.*' indicates all cores in cluster 0 and is equivalent to '0.0.0.0,0.0.0.1,0.0.0.2,...,0.0.0.255').

Base_PowerController - registers

This section describes the registers.

Register summary

This section describes the power control registers in order of offset from the base memory address.

Table 3-12 Base_PowerController register summary

Offset	Name	Type	Reset	Width	Description
0x00	PPOFFR	RW	0x---	32	Power Control Processor Off Register
0x04	PPONR	RW	0x---	32	Power Control Processor On Register
0x08	PCOFFR	RW	0x---	32	Power Control Cluster Off Register
0x0C	PWKUPR	RW	0x---	32	Power Control Wakeup Register
0x10	PSYSR	RW	0x---	32	Power Control SYS Status Register

PPOFFR

The *Power Control Processor Off Register* (PPOFFR) characteristics are: purpose, usage constraints, configurations, and attributes.

Purpose

Processor SUSPEND command when PWKUPR and the GIC are programmed appropriately to provide wakeup events from IRQ and FIQ events to that processor.

Usage constraints

Processor must make power-off requests only for itself.

Configurations

Available in all configurations.

Attributes

See the register summary table.



Figure 3-1 Power Control Processor Off Register bit assignments

Table 3-13 Power Control Processor Off Register bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	ID	MPIDR format affinity value of the processor to be switched off. Programming error if MPIDR != self.

PPONR

The *Power Control Processor On Register* (PPONR) characteristics are: purpose, usage constraints, configurations, and attributes.

Purpose

Brings up a processor from low-power mode.

Usage constraints

Processor must make power-on requests only for other powered-off processors in the system.

Configurations

Available in all configurations.

Attributes

See the register summary table.



Figure 3-2 Power Control Processor On Register bit assignments

Table 3-14 Power Control Processor On Register bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	ID	MPIDR format affinity value of the processor to be switched on. Programming error if MPIDR == self.

PCOFFR

The *Power Control Cluster Off Register* (PCOFFR) characteristics are: purpose, usage constraints, configurations, and attributes.

Purpose

Turns the cluster off.

Usage constraints

Cluster must make power-off requests only for itself.

Configurations

Available in all configurations.

Attributes

See the register summary table.



Figure 3-3 Power Control Cluster Off Register bit assignments

Table 3-15 Power Control Cluster Off Register bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	ID	MPIDR format affinity value of powered-on processor in the cluster to be switched off. Programming error if MPIDR != self.

PWKUPR

The *Power Control Wakeup Register* (PWKUPR) characteristics are: purpose, usage constraints, configurations, and attributes.

Purpose

Configures whether wakeup requests from the GIC are enabled for this cluster.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See the register summary table.

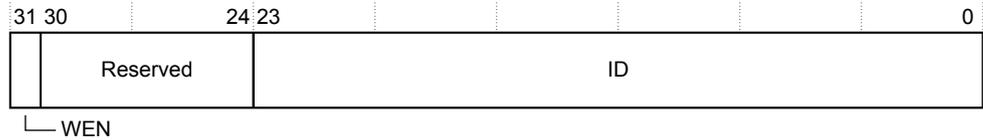


Figure 3-4 Power Control Wakeup Register bit assignments

Table 3-16 Power Control Wakeup Register bit assignments

Bits	Name	Function
[31]	WEN	If set, enables wakeup interrupts (return from SUSPEND) for this cluster.
[30:24]	-	Reserved.
[23:0]	ID	MPIDR format affinity value of processor whose Wakeup Enable bit is to be configured.

PSYSR

The *Power Control SYS Status Register* (PSYSR) characteristics are: purpose, usage constraints, configurations, and attributes.

Purpose

Provides information on the powered status of a given core. Software writes bits [23:0] for the required core and reads the value along with the associated status in bits [31:24].

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See the register summary table.

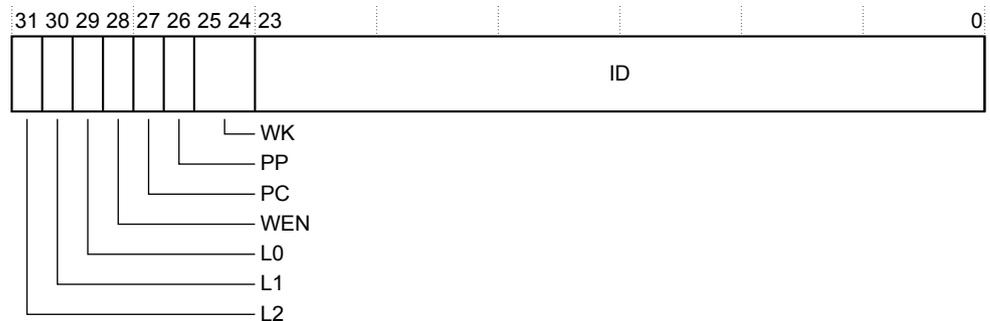


Figure 3-5 Power Control SYS Status Register bit assignments

Table 3-17 Power Control SYS Status Register bit assignments

Bits	Name	Function
[31]	L2	Read-only. A value of 1 indicates that affinity level 2 is active/on. If affinity level 2 is not implemented this bit is RAZ.
[30]	L1	Read-only. A value of 1 indicates that affinity level 1 is active/on. If affinity level 1 is not implemented this bit is RAZ.

Table 3-17 Power Control SYS Status Register bit assignments (continued)

Bits	Name	Function
[29]	L0	Read-only. A value of 1 indicates that affinity level 0 is active/on.
[28]	WEN	Read-only. A value of 1 indicates wakeup interrupts, return from SUSPEND, enabled for this processor. This is an alias of PWKUPR.WEN for this core.
[27]	PC	Read-only. A value of 1 indicates pending cluster off, the cluster enters low-power mode the next time it raises signal STANDBYWFI2.
[26]	PP	Read-only. A value of 1 indicates pending processor off, the processor enters low-power mode the next time it raises signal STANDBYWFI.
[25:24]	WK	Read-only. Indicates the reason for LEVEL0 power on: 0b00 Cold power-on. 0b01 System reset pin. 0b10 Wake by PPNR. 0b11 Wake by GIC WakeRequest signal.
[23:0]	ID	MPIDR format affinity value.

3.8.3 Base - DebugAccessPort component

This section describes the DebugAccessPort component, a model of the *Debug Access Port* (DAP) for external debug connections.

DebugAccessPort - ports

This section describes the ports.

Table 3-18 DebugAccessPort ports

Name	Protocol	Type	Description
ap_pvbus_m[2]	PVBus	Master	Debug-access port to bus master channels 0 and 1.
clock	ClockSignal	Slave	Clock input.
paddrdbg31	Signal	Master	Output signal that indicates which master the access came from, AP0 or AP1. Configurable.

DebugAccessPort - parameters

This section describes the parameters.

Table 3-19 Base Platform DebugAccessPort parameters

Name	Type	Allowed values	Default value	Description
ap0_rom_base_address	uint64_t	0x0-0xffffffffffffffff	0x0	ROM base address for AP0.
ap1_rom_base_address	uint64_t	0x0-0xffffffffffffffff	0x0	ROM base address for AP1.
ap0_has_debug_rom	bool	true, false	false	AP0 has a Debug ROM.
ap1_has_debug_rom	bool	true, false	false	AP1 has a Debug ROM.
ap0_set_paddrdbg31	bool	true, false	false	Set paddrdbg31 signal during accesses on AP0.
ap1_set_paddrdbg31	bool	true, false	false	Set paddrdbg31 signal during accesses on AP1.

3.8.4 Base - simulator visualization component

This section describes the simulator visualization component.

Simulator visualization - parameters

This section describes the parameters.

Table 3-20 Simulator visualization parameters

Parameter	Allowed values	Default value	Description
cluster0_name	-	"Cluster0"	Cluster0 name.
cluster1_name	-	"Cluster1"	Cluster1 name.
cpu_name	-	""	Window title displays core name.
disable_visualisation	true, false	false	Enables or disables visualization.
rate_limit-enable	true, false	true	Rate limit simulation.
recorder.checkInstructionCount	true, false	true	Checks instruction count in recording file against actual instruction count during playback.
recorder.playbackFileName	-	""	Playback filename. An empty string disables playback.
recorder.recordingFileName	-	""	Recording filename. An empty string disables recording.
recorder.recordingTimeBase	-	0x5F5E100	Timebase in 1/s relative to the master clock. For example, 100 000 000 means 10 nanoseconds resolution simulated time for a 1Hz master clock. Used for recording. Higher values give a higher time resolution. Playback time base is always taken from the playback file.
recorder.verbose	-	0x0	Enables verbose messages. 0x0, no messages. 0x1, normal. 0x2, more detail.

Table 3-20 Simulator visualization parameters (continued)

Parameter	Allowed values	Default value	Description
trap_key	-	0x4a	Trap key that works with left Ctrl to toggle mouse display.
window_title	-	"Fast Models - CLCD %cpu%"	cpu_name replaces window title, %cpu%.

3.8.5 Base - VE_SysRegs component

This section describes the VE system registers component.

VE_SysRegs - parameters

This section describes the parameters.

Table 3-21 Base Platform VE_SysRegs parameters

Name	Type	Allowed values	Default value	Description
exit_on_shutdown	bool	true, false	false	true: if software uses the SYS_CFGCTRL function SYS_CFG_SHUTDOWN, then the simulator shuts down and exits. ^h
mmbSiteDefault	uint8_t	0x0-0x2	0x1	Default MMB source. 0x0, MB. 0x1, DB1. 0x2, DB2.
tilePresent	bool	true, false	true	Tile fitted.
user_switches_value	uint32_t	-	0x0	User switches.

VE_SysRegs - registers

This section describes the configuration registers.

Table 3-22 Base Platform VE_SysRegs registers

Name	Offset	Access	Description
SYS_ID	0x00	Read/write	System identity.
SYS_SW	0x04	Read/write	Bits[7:0] map to switch S6.
SYS_LED	0x08	Read/write	Bits[7:0] map to user LEDs.
SYS_100HZ	0x24	Read only	100Hz counter.
SYS_FLAGS	0x30	Read/write	General-purpose flags.
SYS_FLAGSCLR	0x34	Write only	Clear bits in general-purpose flags.
SYS_NVFLAGS	0x38	Read/write	General-purpose non-volatile flags.
SYS_NVFLAGSCLR	0x3C	Write only	Clear bits in general-purpose non-volatile flags.
SYS_MCI	0x48	Read only	MCI.
SYS_FLASH	0x4C	Read/write	Flash control.
SYS_CFGSW	0x58	Read/write	Boot-select switch.
SYS_24MHZ	0x5C	Read only	24MHz counter.

^h For more information on SYS_CFGCTRL, see the *Motherboard Express μATX V2M-P1 Technical Reference Manual*.

Table 3-22 Base Platform VE_SysRegs registers (continued)

Name	Offset	Access	Description
SYS_MISC	0x60	Read/write	Miscellaneous control flags.
SYS_DMA	0x64	Read/write	DMA peripheral map.
SYS_PROCID0	0x84	Read/write	Processor ID.
SYS_PROCID1	0x88	Read/write	Processor ID.
SYS_PROCID2	0x8C	Read/write	Processor ID.
SYS_PROCID3	0x90	Read/write	Processor ID.
SYS_CFGDATA	0xA0	Read/write	Data to read/write from & to motherboard controller.
SYS_CFGCTRL	0xA4	Read/write	Control data transfer to motherboard controller.
SYS_CFGSTAT	0xA8	Read/write	Status of data transfer to motherboard controller.

3.9 Base - differences between the AEMv8-A FVP and core FVPs

This section describes implementation features of the core models that the AEMv8-A model does not implement, or implements with significant differences.

- The default value of `cache_state_modelled` is 0.
- Components `cluster0` and `cluster1` are implementation cores, not AEMs. All parameters in these components are the parameters of the core implementation, not the parameters of the AEM. The values of `bp.proc_id0` and `bp.proc_id1` have fixed values consistent with the cores and are not configurable.
- The core defines the memory map of register banks within the GIC region, and the map is therefore not configurable. The parameter `bp.variant` communicates the nature of the memory map to target software. The parameter has a fixed value consistent with the memory map of the core, and is not configurable. Because the core implementations contain a specific version of the GIC, the parameter `gicv3.gicv2-only` is not available. The following registers in the GIC distributor are given fixed values to match the implementation, and are not configurable: `gic_distributor.reg-base`, `gic_distributor.reg-base-per-distributor`, `gic_distributor.GICD-alias`, `gic_distributor.ITS0-base`.
- The GICv2 CPU IDs are contiguous in implementation platform models, because the number of cores in each cluster is fixed. They can be non-contiguous in the AEMv8-A Base Platform FVP because there is space for four cores in the cluster. You can configure fewer than four cores.

3.10 Base - VE compatibility

Arm expects software that ran on the previous VE model to be compatible with this system model, but you might need to apply some configuration options.

This section contains the following subsections:

- [3.10.1 Base - VE compatibility - GICv2 on page 3-47.](#)
- [3.10.2 Base - VE compatibility - GICv3 on page 3-47.](#)
- [3.10.3 Base - VE compatibility - system global counter on page 3-48.](#)
- [3.10.4 Base - VE compatibility - disable security on page 3-48.](#)

3.10.1 Base - VE compatibility - GICv2

This system model uses GICv3 by default. You can configure it to support GICv2 or GICv2m.

To configure the model as GICv2m, set the following:

```
-C gicv3.gicv2-only=1 \
-C cluster0.gic.GICD-offset=0x1000 \
-C cluster0.gic.GICC-offset=0x2F000 \
-C cluster0.gic.GICH-offset=0x4F000 \
-C cluster0.gic.GICH-other-CPU-offset=0x50000 \
-C cluster0.gic.GICV-offset=0x6F000 \
-C cluster0.gic.PERIPH-size=0x80000 \
-C cluster1.gic.GICD-offset=0x1000 \
-C cluster1.gic.GICC-offset=0x2F000 \
-C cluster1.gic.GICH-offset=0x4F000 \
-C cluster1.gic.GICH-other-CPU-offset=0x50000 \
-C cluster1.gic.GICV-offset=0x6F000 \
-C cluster1.gic.PERIPH-size=0x80000 \
-C gic_distributor.GICD-alias=0x2c010000
```

To configure the model as GICv2, set the following:

```
-C gicv3.gicv2-only=1 \
-C cluster0.gic.GICD-offset=0x1000 \
-C cluster0.gic.GICC-offset=0x2000 \
-C cluster0.gic.GICH-offset=0x4000 \
-C cluster0.gic.GICH-other-CPU-offset=0x5000 \
-C cluster0.gic.GICV-offset=0x6000 \
-C cluster0.gic.PERIPH-size=0x8000 \
-C cluster1.gic.GICD-offset=0x1000 \
-C cluster1.gic.GICC-offset=0x2000 \
-C cluster1.gic.GICH-offset=0x4000 \
-C cluster1.gic.GICH-other-CPU-offset=0x5000 \
-C cluster1.gic.GICV-offset=0x6000 \
-C cluster1.gic.PERIPH-size=0x8000 \
-C gic_distributor.GICD-alias=0x2c010000
```

To configure MSI frames for GICv2m, parameters are available to set the base address and configuration of each of 16 possible frames. Eight frames are Secure and eight frames are Non-secure:

```
-C gic_distributor.MSI_S-frame0-base=ADDRESS \
-C gic_distributor.MSI_S-frame0-min-SPI=NUM \
-C gic_distributor.MSI_S-frame0-max-SPI=NUM
```

In this example, you can replace `MSI_S` with `MSI_NS`, for NS frames, and you can replace `frame0` with `frame1` to `frame7` for each of the possible 16 frames. If the base address is not specified for a given frame, or the SPI numbers are out of range, the corresponding frame is not instantiated.

3.10.2 Base - VE compatibility - GICv3

If a Base Platform includes an implementation of the GICv3 system registers, it is enabled by default.

The GIC distributor and CPU (core) interface have parameters that allow configuration of the model to match different implementation options. Use `--list-params` to get a full list. Configuration options for the GIC model must be available under:

- `cluster[0-n].gic.*`
- `cluster[0-n].gicv3.*`
- `gic_distributor.*`

3.10.3 Base - VE compatibility - system global counter

The Generic Timer registers of the cores do not operate by default.

The model provides a memory-mapped interface to the system global counter, and enables the free-running timer from reset. However, the architectural requirement is that such a counter is not enabled at reset. As a result, the Generic Timer registers of the cores do not operate unless either:

- Software enables the counter peripheral by writing the FCREQ[0] and EN bits in CNTCR at 0x2a43000. Arm recommends this approach.
- The -C bp.refcounter.non_arch_start_at_default=1 parameter is set. This approach provides compatibility with older software.

3.10.4 Base - VE compatibility - disable security

Base Platform FVPs have an enhanced security map for peripherals. By default, it restricts access to some peripherals.

Software must program the TZC-400 to make any accesses to DRAM, because the reset configuration blocks all accesses.

For backward compatibility with software that cannot program the TZC-400, this parameter setting permits all accesses regardless of security state:

```
-C bp.secure_memory=false
```

3.11 Base - unsupported VE features

This system model does not support software that relies on some features of the VE model.

This section contains the following subsections:

- [3.11.1 Base - unsupported VE features - memory aliasing at 0x08_00000000](#) on page 3-49.
- [3.11.2 Base - unsupported VE features - boot ROM alias at 0x00_0800_0000](#) on page 3-49.
- [3.11.3 Base - unsupported VE features - change of older parameters](#) on page 3-49.

3.11.1 Base - unsupported VE features - memory aliasing at 0x08_00000000

The VE model permits an alias of the 2GB region of DRAM between addresses 0x80000000 and 0xFFFFFFFF with addresses 0x08_00000000 to 0x08_7FFFFFFF. The Base Platform does not have this alias and the region 0x08_00000000 to 0x08_7FFFFFFF is Reserved.

3.11.2 Base - unsupported VE features - boot ROM alias at 0x00_0800_0000

In the VE model, the region at 0x00_0800_0000 was an alias of the trusted boot ROM at 0x00_0000_0000. It is now an independent region of NOR flash.

3.11.3 Base - unsupported VE features - change of older parameters

Most parameter names have been simplified between the VE model and the Base Platform system model.

Components that were previously in *motherboard* or *daughterboard* groups are now in a *bp* group. The model does not recognize the previous parameter names.

In a change to the previous default, the Base Platform models the core cache state by default. You can disable this using a single parameter for all cores in the simulation, using the `cache_state_modelled` parameter.

```
-C cache_state_modelled=0
```

————— **Note** —————

Cortex Base Platforms do not model the cache state by default.

Chapter 4

Programming Reference for MPS2 FVPs

This chapter describes the model of the hardware platform.

It contains the following sections:

- *4.1 MPS2 - about* on page 4-51.
- *4.2 MPS2 platform types* on page 4-52.
- *4.3 MPS2 - memory maps* on page 4-53.
- *4.4 MPS2 - interrupt assignments* on page 4-59.
- *4.5 MPS2 - differences between models and hardware* on page 4-60.

4.1 MPS2 - about

The *Microcontroller Prototyping System 2 (MPS2) Fixed Virtual Platform (FVP)* model implements a subset of the functionality of the V2M-MPS2/V2M-MPS2+ motherboard hardware.

MPS2 model platforms include MPS2 components and generic ones, such as buses and timers.

MPS2 platforms are sufficiently accurate to boot the Keil® RTX RTOS and run the Blinky application.

To list the model parameters and their types, allowed values, default values, and descriptions, run the model with the `--list-params` argument.

Some MPS2 example systems are available in the `%PVLIB_HOME%\examples\LISA\FVP_MPS2\` directory.

Related information

AN400 - Arm Cortex-M7 SMM on V2M-MPS2

4.2 MPS2 platform types

Configure the MPS2 FVP platform type using the `fvmp2.platform_type` parameter.

It has the following possible values:

- 0 The FVP acts as a V2M-MPS2 system, with the additions for v8-M, as specified in the Armv8-M MPS2 System Specification (ECM 0468897), v0.8. This specification is confidential and is available only to licensed Arm customers. For details, contact your Arm support representative. This value is the default.
- 1 The FVP acts as an IoT Kit on an MPS2+ board. For details, see the following documents:
 - Cortex-M23 processor Armv8-M IoT Kit User Guide (ECM 0635473), <http://infocenter.arm.com/help/topic/com.arm.doc.ecm0635473/index.html>.
 - Cortex-M33 processor Armv8-M IoT Kit User Guide (ECM 0601256), <http://infocenter.arm.com/help/topic/com.arm.doc.ecm0601256/index.html>.
- 2 The FVP acts as an Arm CoreLink SSE-200 Subsystem on an MPS2+ board. For details, see AN521 - SMM Cortex-M33 SSE-200 for MPS2+ Application Note, <http://infocenter.arm.com/help/topic/com.arm.doc.dai0521a/index.html>.

4.3 MPS2 - memory maps

This section describes the MPS2 memory maps.

This section contains the following subsections:

- [4.3.1 MPS2 - memory map for models without the Armv8-M additions](#) on page 4-53.
- [4.3.2 MPS2 - memory map for models with the Armv8-M additions](#) on page 4-54.

4.3.1 MPS2 - memory map for models without the Armv8-M additions

This section describes the MPS2 memory map for older cores, without the Armv8-M additions.

For standard Arm peripherals, see the TRM for that device.

————— **Note** —————

- A bus error is generated for accesses to memory areas not shown in this table.
- Any memory device that does not occupy the total region is aliased within that region.

Table 4-1 Overview of MPS2 memory map

Description	Modeled	Address range
Ethernet ⁱ	Partial	0xA0000000-0xA000FFFF
PSRAM (16MB)	Yes	0x60000000-0x60FFFFFF
VGA Image (512x128) (AHB)	Yes	0x41100000-0x4110FFFF
VGA Console (AHB)	Yes	0x41000000-0x4100FFFF
Block RAM (boot time) ^j	Yes	0x40200000-0x402FFFFFF
Reserved	N/A	0x40030000-0x401FFFFFF
SCC register	Yes	0x4002F000-0x4002FFFF
Reserved	N/A	0x40029000-0x4002EFFF
FPGA System Control & I/O, APB	Yes	0x40028000-0x40028FFF
Reserved	N/A	0x40025000-0x40027FFF
Audio I2S, APB	Partial	0x40024000-0x40024FFF
SBCon (Audio Configuration), APB	Yes	0x40023000-0x40023FFF
SBCon (Touch for LCD module), APB	Partial	0x40022000-0x40022FFF
PL022 (SPI for LCD module), APB	Partial	0x40021000-0x40021FFF
PL022 (SPI), APB	Yes	0x40020000-0x40020FFF
CMSDK system controller	Yes	0x4001F000-0x4001FFFF
Reserved for extra GPIO & other AHB peripherals	N/A	0x40012000-0x4001EFFF
CMSDK AHB GPIO #1	Yes	0x40011000-0x40011FFF
CMSDK AHB GPIO #0	Yes	0x40010000-0x40010FFF
CMSDK APB subsystem	Yes	0x40000000-0x4000FFFF
Reserved	N/A	0x20800000-0x20FFFFFF

ⁱ Through ahb_to_extmem16. Offset 0x0-0x0FE for CSRs, 0x100-0x1FE for FIFO.

^j Reserved 64KB, 16K implemented. This memory is wrapped through the region.

Table 4-1 Overview of MPS2 memory map (continued)

Description	Modeled	Address range
ZBTSRAM 2 & 3 (2x32-bit) ^k	Yes	0x20000000-0x207FFFFFFF
Reserved	N/A	0x01010000-0x1FFFFFFF
Reserved	N/A	0x00800000-0x00FFFFFF
ZBTSRAM 1 (64-bit) ^l	Yes	0x00400000-0x007FFFFFFF
ZBTSRAM 1 (64-bit)	Yes	0x00004000-0x003FFFFFFF
Mappable memory ^m	Yes	0x00000000-0x00003FFF

4.3.2 MPS2 - memory map for models with the Armv8-M additions

This section describes the MPS2 memory map for newer cores, with the Armv8-M additions.

For standard Arm peripherals, see the TRM for that device.

Note

- A bus error is generated for accesses to memory areas not shown in this table.
- Any memory device that does not occupy the total region is aliased within that region.

Table 4-2 Overview of MPS2 memory map

Description	IDAU	Modeled	Address range
ZBTSRAM 1 (4MB) in <i>Non-secure</i> (NS) world. Reserved 8MB, only 4MB implemented. VTOR initialization value to be configurable in LAC). Second half (4MB) aliased to first half (4MB).	NS	Yes	0x00000000-0x007FFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x00800000-0x0FFFFFFF
ZBTSRAM 1 (4MB) in <i>Secure</i> (S) world. Reserved 8MB, only 4MB implemented. Second half (4MB) aliased to first half (4MB).	S	Yes	0x10000000-0x107FFFFFFF
Not used. Default expansion port: bus error.	S	N/A	0x10800000-0x1FFFFFFF
ZBTSRAM 2&3 (4MB) in NS world. Reserved 8MB, only 4MB implemented. For IoT subsystems, different cores have different memory sizes. Second half (4MB) aliased to first half (4MB).	NS	Yes	0x20000000-0x207FFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x20800000-0x20FFFFFFF
PSRAM (16MB)	NS	Yes	0x21000000-0x21FFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x22000000-0x23FFFFFFF
MTB SRAM. Reserved 64KB, only 16KB implemented. Aliased to 0x0 for booting in RTL simulation.	NS	Yes	0x24000000-0x2400FFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x24010000-0x24FFFFFFF

^k Reserved 8MB, 4MB available. The two SRAM blocks are interleaved.

^l Wrapped. Only 4MB ZBTSRAM fitted.

^m When `zbt_boot_ctrl1 = 0`, ZBTSRAM 1 is mapped to this region. Otherwise, `Remap_ctrl1 = 0` maps Block RAM and `Remap_ctrl1 = 1` maps ZBTSRAM 1. The V2M-MPS2 board microcontroller controls the `zbt_boot_ctrl` signal. The `zbt_boot_ctrl` signal overrides the boot option to enable use of the ZBT RAM.

Table 4-2 Overview of MPS2 memory map (continued)

Description	IDAU	Modeled	Address range
ZBTSRAM 2&3 (4MB) in S world. Reserved 8MB, only 4MB implemented. Second half (4MB) aliased to first half (4MB).	S	Yes	0x30000000-0x307FFFFFFF
Not used. Default expansion port: bus error.	S	N/A	0x30800000-0x30FFFFFFF
Not used. No memory gating unit on PSRAM (16MB) path because it is shared with Ethernet control. Default expansion port: bus error.	S	N/A	0x31000000-0x31FFFFFFF
Not used. Default expansion port: bus error.	S	N/A	0x30800000-0x3FFFFFFF
CMSDK APB subsystem:	NS	-	0x40000000-0x4000FFFF
Timer 0.	NS	Yes	0x40000000-0x40000FFF
Timer 1.	NS	Yes	0x40001000-0x40001FFF
Dual Timer.	NS	Yes	0x40002000-0x40002FFF
Not used.	NS	N/A	0x40003000-0x40003FFF
UART #0.	NS	Yes	0x40004000-0x40004FFF
UART #1.	NS	Yes	0x40005000-0x40005FFF
UART #2.	NS	Yes	0x40006000-0x40006FFF
Not used.	NS	N/A	0x40007000-0x40007FFF
Watchdog.	NS	Yes	0x40008000-0x40008FFF
Not used.	NS	N/A	0x40009000-0x4000F000
GPIO #0.	NS	Yes	0x40010000-0x40010FFF
GPIO #1.	NS	Yes	0x40011000-0x40011FFF
GPIO #2.	NS	Yes	0x40012000-0x40012FFF
GPIO #3.	NS	Yes	0x40013000-0x40013FFF
Default slave inside AHB peripheral subsystem. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	Yes	0x40014000-0x40017FFF
DMA Controller #0.	NS	Yes	0x40018000-0x40018FFF
DMA Controller #1.	NS	Yes	0x40019000-0x40019FFF
Default slave inside AHB peripheral subsystem. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	Yes	0x4001A000-0x40017FFF
CMSDK system controller. PMU control and remap registers unused. Only reset option (lockup reset) and rest info available.	NS	Yes	0x4001F000-0x4001FFFF
FPGA APB Subsystem (unused APB space: RAZ/WI):	NS	-	0x40020000-0x4002FFFF
PL022 (SPI).	NS	Yes	0x40020000-0x40020FFF
PL022 (SPI for LCD).	NS	Partial	0x40021000-0x40021FFF
SBCon I2C (Touch for LCD).	NS	Partial	0x40022000-0x40022FFF
SBCon I2C (Audio configuration).	NS	Yes	0x40023000-0x40023FFF

Table 4-2 Overview of MPS2 memory map (continued)

Description	IDAU	Modeled	Address range
Audio I2S.	NS	Partial	0x40024000-0x40024FFF
Not used.	NS	N/A	0x40025000-0x40027FFF
FPGA system control & I/O (LEDs, buttons...).	NS	Yes	0x40028000-0x40028FFF
Not used.	NS	N/A	0x40029000-0x4002EFFF
SCC registers.	NS	Yes	0x4002F000-0x401FFFFF
Ethernet.	NS	Partial	0x40200000-0x402FFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x40300000-0x40FFFFFF
VGA console.	NS	Yes	0x41000000-0x4100FFFF
VGA image.	NS	Yes	0x41100000-0x4113FFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x41140000-0x4FFFFFFF
CMSDK APB subsystem:	S	-	0x50000000-0x5000FFFF
Timer 0.	S	Yes	0x50000000-0x50000FFF
Timer 1.	S	Yes	0x50001000-0x50001FFF
Dual Timer.	S	Yes	0x50002000-0x50002FFF
Not used.	S	N/A	0x50003000-0x50003FFF
UART #0.	S	Yes	0x50004000-0x50004FFF
UART #1.	S	Yes	0x50005000-0x50005FFF
UART #2.	S	Yes	0x50006000-0x50006FFF
Not used.	S	N/A	0x50007000-0x50007FFF
Watchdog.	S	Yes	0x50008000-0x50008FFF
Not used.	S	N/A	0x50009000-0x5000F000
GPIO #0	S	Yes	0x50010000-0x50010FFF
GPIO #1	S	Yes	0x50011000-0x50011FFF
GPIO #2	S	Yes	0x50012000-0x50012FFF
GPIO #3	S	Yes	0x50013000-0x50013FFF
Default slave. Default expansion port: bus error.	S	Yes	0x50014000-0x50017FFF
DMA Controller #0.	S	Yes	0x50018000-0x50018FFF
DMA Controller #1.	S	Yes	0x50019000-0x50019FFF
Default slave. Default expansion port: bus error.	S	Yes	0x5001A000-0x5001EFFF
CMSDK system controller. PMU control and remap registers unused. Only reset option (lockup reset) and rest info available.	S	Yes	0x5001F000-0x5001FFFF

Table 4-2 Overview of MPS2 memory map (continued)

Description	IDAU	Modeled	Address range
FPGA APB subsystem:	S	-	0x50020000-0x5002FFFF
PL022 (SPI).	S	Yes	0x50020000-0x50020FFF
PL022 (SPI for LCD).	S	Partial	0x50021000-0x50021FFF
SBCon I2C (touch for LCD).	S	Partial	0x50022000-0x50022FFF
SBCon I2C (audio configuration).	S	Yes	0x50023000-0x50023FFF
Audio I2S.	S	Partial	0x50024000-0x50024FFF
Not used.	S	N/A	0x50025000-0x50027FFF
FPGA system control & I/O (LEDs, buttons...).	S	Yes	0x50028000-0x50028FFF
Not used.	S	N/A	0x50029000-0x5002EFFF
SCC registers.	S	Yes	0x5002F000-0x501FFFFFFF
Ethernet.	S	Partial	0x50200000-0x502FFFFFFF
Not used. Default expansion port: bus error.	S	N/A	0x50300000-0x50FFFFFFF
VGA console.	S	Yes	0x51000000-0x510FFFFFFF
VGA image.	S	Yes	0x51100000-0x5113FFFFFF
Not used. Default expansion port: bus error.	S	N/A	0x51140000-0x57FFFFFFFS
Secure APB subsystem:	S	-	0x58000000-0x5800FFFF
TRNG / PRNG.	S	Yes	0x58000000-0x58000FFF
Unique ID or Secure storage.	S	Yes	0x58001000-0x58006FFF
Secure Control Registers.	S	Yes	0x58007000-0x58007FFF
Flash memory gating unit configuration (mapped to AHB port for CODE region in the bus matrix, not APB).	S	Yes	0x58008000-0x58009FFF
SRAM memory gating unit configuration (mapped to AHB port for SRAM region in the bus matrix, not APB).	S	Yes	0x5800A000-0x5800DFFF
Reserved.	S	N/A	0x5800E000-0x5800EFFF
Reserved.	S	N/A	0x5800F000-0x5800FFFF
Not used. Default expansion port: bus error.	S	N/A	0x50010000-0x5FFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x60000000-0x6FFFFFFF
Not used. Default expansion port: bus error.	S	N/A	0x70000000-0x7FFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0x80000000-0x8FFFFFFF
Not used. Default expansion port: bus error.	S	N/A	0x90000000-0x9FFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0xA0000000-0xAFFFFFFF

Table 4-2 Overview of MPS2 memory map (continued)

Description	IDAU	Modeled	Address range
Not used. Default expansion port: bus error.	S	N/A	0xB0000000-0xBFFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0xC0000000-0xCFFFFFFF
Not used. Default expansion port: bus error.	S	N/A	0xD0000000-0xDFFFFFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0xE0000000-0xEFFFFFFF
System ROM table. Exempted from checking.	Exempt	Yes	0xF0000000-0xF0000FFF
Not used. Default expansion port: bus error.	Exempt	N/A	0xF0001000-0xF000FFFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0xF0100000-0xF0100FFF
MTB SFR address space.	NS	Yes	0xF0200000-0xF0200FFF
Reserved. This region is non-executable.	NS	N/A	0xF0210000-0xF0213FFF
Not used. Default expansion port (MPS2 AHB subsystem): bus error or RAZ/WI.	NS	N/A	0xF0214000-0xFFFFFFFF

4.4 MPS2 - interrupt assignments

This section describes the interrupt assignments.

Table 4-3 Interrupt assignments

Number	Interrupt
NMI	Watchdog.
0	UART 0 receive interrupt.
1	UART 0 transmit interrupt.
2	UART 1 receive interrupt.
3	UART 1 transmit interrupt.
4	UART 2 receive interrupt.
5	UART 2 transmit interrupt.
6	GPIO 0, 2 combined interrupt.
7	GPIO 1, 3 combined interrupt.
8	Timer 0.
9	Timer 1.
10	Dual Timer.
11	SPI #1 (LCD). The LCD had shared SPI #0 and SPI #1.
12	UART overflow (0, 1, 2).
13	Ethernet.
14	Audio I2S.
15	Touch screen.
16-31	GPIO 0 individual interrupts.
32-47	GPIO 1 individual interrupts. Armv8-M additions.
48	SPI #0. Armv8-M addition.
49	Reserved.
50	TRNG (Secure). Armv8-M addition.
51	Unique ID and Secure storage (Secure). Armv8-M addition.
52	DMA controller #0.
53	DMA controller #1.
54	SecureErrorIRQ. Armv8-M addition. ⁿ

ⁿ Detection of Non-secure access to Secure address spaces (including other bus masters). Generated by Memory Gating unit, Peripheral Gating units, bus gasket for legacy bus masters.

4.5 MPS2 - differences between models and hardware

This section describes the features of the hardware that the models do not implement, or implement with significant differences.

MPS2 implements most devices. Some peripherals have minimal implementations:

- The Ethernet module in the model is a LAN91C111. The hardware provides a LAN9220.
- The Audio module is RAZ/WI.
- The STMPE811 touchscreen module only reports touch positions.
- The model of the Ampire LCD module supports a subset of the graphics modes.

You can display images and text on an emulated VGA output, images on the LCD, and text on the UART.

Armv8-M

The model does not support MTB, ETM, and TPIU. MTB RAM is absent.

In the Memory Gating Unit, the model provides a configurable block size. For performance reasons, the minimum block size in the model is 4096 bytes. Hardware and later models might allow smaller block sizes. Software must use the BLK_CFG register to determine block size.

Timing

FVPs enable software applications to run in a functionally accurate simulation. However, because of the relative balance of fast simulation speed over timing accuracy, there are situations where the models might behave unexpectedly.

If your code interacts with real world devices such as timers and keyboards, data arrives in the modeled device in real world, or wall clock, time. However, simulation time can run faster than the wall clock. So, a single key press might be interpreted as several repeated key presses, or a single mouse click might be interpreted as a double click.

To avoid this mismatch, the FVPs provide the Rate Limit feature. Enabling Rate Limit forces the model to run at wall clock time. For interactive applications, Arm recommends enabling Rate Limit. Use the Rate Limit button in the CLCD display or the `rate_limit-enable` model instantiation parameter.

Chapter 5

Programming Reference for VE FVPs

This chapter describes the memory map and the parameters for the peripheral and system component models.

It contains the following sections:

- *5.1 VE - about* on page 5-62.
- *5.2 Memory maps for VE FVPs* on page 5-64.
- *5.3 Interrupt maps for VE FVPs* on page 5-68.
- *5.4 VE parameters* on page 5-70.
- *5.5 VE - components* on page 5-72.
- *5.6 Differences between the VE hardware and the system model* on page 5-78.

5.1 VE - about

The Versatile Express (VE) FVPs are functionally accurate system models for software execution. A range of VE FVPs are supplied as standalone products and as examples in Fast Models.

Arm produces the VE hardware development platform. The Motherboard Express *μAdvanced Technology Extended* (ATX) V2M-P1 is the basis for an integrated software and hardware development system. This system is also based on the Arm *Symmetric MultiProcessor* (SMP) system architecture.

The VE FVPs are system models implemented in software. Each model contains:

- A virtual implementation of the Arm VE motherboard.
- A single daughterboard containing one or more Arm processors.
- Associated interconnections.

The motherboard provides:

- Peripherals for multimedia or networking environments.
- Access to motherboard peripherals and functions through a static memory bus to simplify access from daughterboards.
- High-performance PCI-Express slots for expansion cards.
- Consistent memory maps with different processor daughterboards that simplify software development and porting.
- Automatic detection and configuration of attached CoreTile Express and LogicTile Express daughterboards.
- Automatic shutdown for over-temperature or power supply failure.
- No system power-on for unconfigurable daughterboards.
- Power sequencing of system.
- Drag and drop file updating of configuration files.
- Support of either a 12V power-supply unit or an external ATX power supply.
- Support of FPGA and processor daughterboards to provide custom peripherals, early access to processor designs, or production test chips.

Note

Arm bases the models on the VE platform memory map, but does not intend them to be accurate representations of a specific VE hardware revision. The VE FVPs support selected peripherals. The models are sufficiently complete and accurate to boot the same operating system images as the VE hardware.

VE FVPs provide functionally accurate models for software execution. However, the models sacrifice timing accuracy to increase simulation speed. Key deviations from hardware are:

- Approximate timing.
- Simplified buses.
- No implementations for processor caches and the related write buffers.

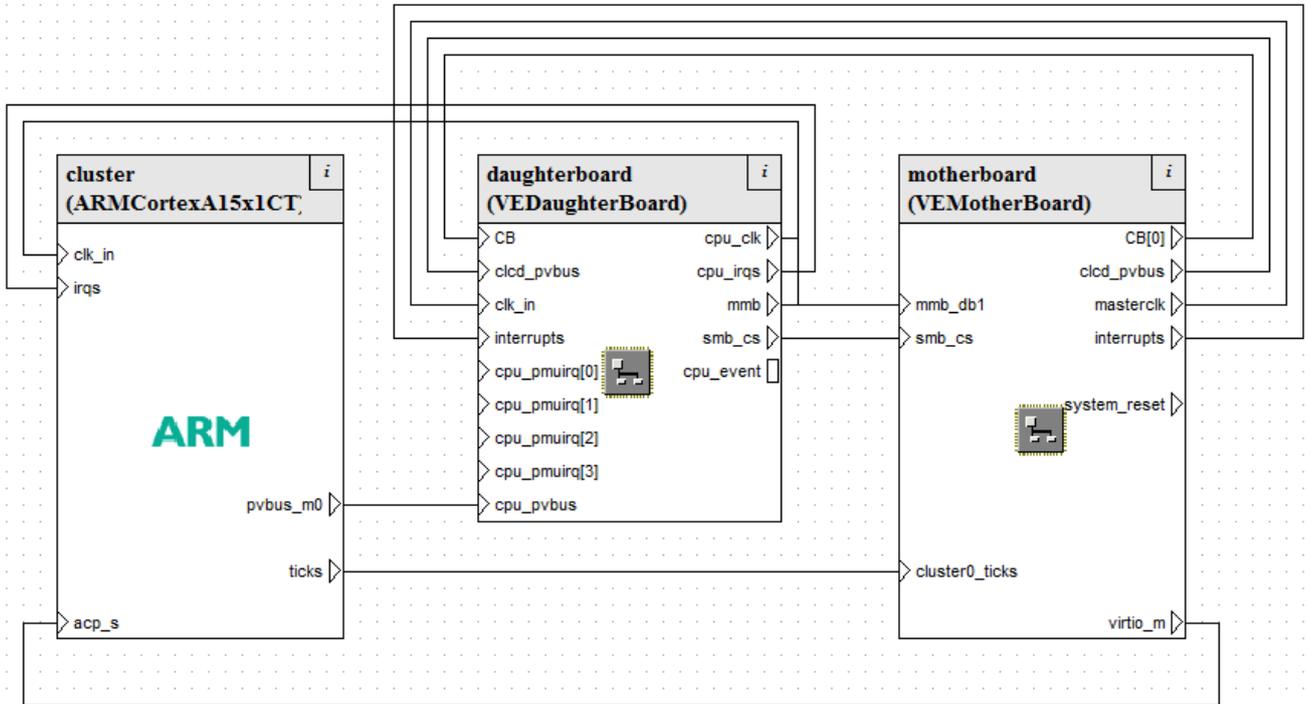


Figure 5-1 Top-level block diagram of a VE model

Related information

Motherboard Express μ ATX V2M-PI Technical Reference Manual

5.2 Memory maps for VE FVPs

This section describes the memory maps for the VE FVPs.

This section contains the following subsections:

- [5.2.1 VE memory map for Cortex®-A series on page 5-64.](#)
- [5.2.2 VE memory map for Cortex®-R series on page 5-66.](#)

5.2.1 VE memory map for Cortex®-A series

The global memory map for the Cortex-A series VE model is based on the hardware Versatile Express RS1 memory map with the RS2 extensions.

————— **Note** —————

The VE FVP implementation of memory does not require the memory controller to have the correct values. If you run applications on hardware, ensure that the memory controller is set up properly. Otherwise, applications that run on the FVP might fail on hardware.

Table 5-1 Cortex-A series platform model memory map

Name	Modeled	Address range	Size
NOR FLASH0 (CS0)	Yes	0x00_00000000-0x00_03FFFFFF	64MB
Reserved	-	0x00_04000000-0x00_07FFFFFF	64MB
NOR FLASH0 alias (CS0)	Yes	0x00_08000000-0x00_0BFFFFFF	64MB
NOR FLASH1 (CS4)	Yes	0x00_0C000000-0x00_0FFFFFFF	64MB
Unused (CS5)	-	0x00_10000000-0x00_13FFFFFF	-
PSRAM (CS1) - unused	No	0x00_14000000-0x00_17FFFFFF	-
Peripherals (CS2). See CS2 region peripheral memory map on page 5-65 , below.	Yes	0x00_18000000-0x00_1BFFFFFF	64MB
Peripherals (CS3). See CS3 region peripheral memory map on page 5-65 , below.	Yes	0x00_1C000000-0x00_1FFFFFFF	64MB
CoreSight and peripherals	No	0x00_20000000-0x00_2CFFFFFF ^o	-
Graphics space	No	0x00_2D000000-0x00_2D00FFFF	-
System SRAM	Yes	0x00_2E000000-0x00_2EFFFFFF	64KB
Ext AXI	No	0x00_2F000000-0x00_7FFFFFFF	-
4GB DRAM (in 32-bit address space) ^p	Yes	0x00_80000000-0x00_FFFFFFFF	2GB
Unused	-	0x01_00000000-0x07_FFFFFFFF	-
4GB DRAM (in 36-bit address space) ^p	Yes	0x08_00000000-0x08_FFFFFFFF	4GB
Unused	-	0x09_00000000-0x7F_FFFFFFFF	-
4GB DRAM (in 40-bit address space) ^p	Yes	0x80_00000000-0xFF_FFFFFFFF	4GB

^o The private peripheral region address 0x2c000000 is mapped in this region. You can use the parameter PERIPBASE to map the peripherals to a different address.
^p The model contains a single 4GB block of DRAM, which is aliased across the three different regions. In other words, it can be accessed at three different physical addresses, which are all mapped to the same area of DRAM. For example, a write to address 0x00_80000000 will be visible to reads at address 0x80_00000000. The lowest of the physical address regions is only 2GB in size.

The model has a `secure_memory` option. When you enable this option, the memory map changes for a number of peripherals.

Table 5-2 CS2 region peripheral memory map for `secure_memory` option

Peripheral	Address range	Functionality with <code>secure_memory</code> enabled
NOR FLASH0 (CS0)	0x00_00000000-0x00_0001FFFF	Secure RO, aborts on non-secure accesses.
Reserved	0x00_04000000-0x00_0401FFFF	Secure SRAM, aborts on non-secure accesses.
NOR FLASH0 alias (CS0)	0x00_08000000-0x00_7DFFFFFF	Normal memory map, aborts on secure accesses.
Ext AXI	0x00_7e000000-0x00_7FFFFFFF	Secure DRAM, aborts on non-secure accesses.
4GB DRAM (in 32-bit address space)	0x00_80000000-0xFF_FFFFFFFF	Normal memory map, aborts on secure accesses.

Table 5-3 CS2 region peripheral memory map

Peripheral	Modeled	Address range	Size	GIC Int ^q
VRAM - aliased	Yes	0x00_18000000-0x00_19FFFFFF	32MB	-
Ethernet (SMSC 91C111)	Yes	0x00_1A000000-0x00_1AFFFFFF	16MB	47
USB - unused	No	0x00_1B000000-0x00_1BFFFFFF	16MB	-

Table 5-4 CS3 region peripheral memory map

Peripheral	Modeled	Address range	Size	GIC Int ^q
Local DAP ROM	No	0x00_1C000000-0x00_1C00FFFF	64KB	-
VE System Registers	Yes	0x00_1C010000-0x00_1C01FFFF	64KB	-
System Controller (SP810)	Yes	0x00_1C020000-0x00_1C02FFFF	64KB	-
TwoWire serial interface (PCIe)	No	0x00_1C030000-0x00_1C03FFFF	64KB	-
AACI (PL041)	Yes	0x00_1C040000-0x00_1C04FFFF	64KB	43
MCI (PL180)	Yes	0x00_1C050000-0x00_1C05FFFF	64KB	41, 42
KMI - keyboard (PL050)	Yes	0x00_1C060000-0x00_1C06FFFF	64KB	44
KMI - mouse (PL050)	Yes	0x00_1C070000-0x00_1C07FFFF	64KB	45
Reserved	-	0x00_1C080000-0x00_1C08FFFF	64KB	-
UART0 (PL011)	Yes	0x00_1C090000-0x00_1C09FFFF	64KB	37
UART1 (PL011)	Yes	0x00_1C0A0000-0x00_1C0AFFFF	64KB	38
UART2 (PL011)	Yes	0x00_1C0B0000-0x00_1C0BFFFF	64KB	39
UART3 (PL011)	Yes	0x00_1C0C0000-0x00_1C0CFFFF	64KB	40
Reserved	-	0x00_1C0D0000-0x00_1C0EFFFF	128KB	-
Watchdog (SP805)	Yes	0x00_1C0F0000-0x00_1C0FFFFFF	64KB	32
Reserved	-	0x00_1C100000-0x00_1C10FFFF	64KB	-
Timer-0 (SP804)	Yes	0x00_1C110000-0x00_1C11FFFF	64KB	34

^q Use these interrupt signal values to program your interrupt controller. They are the SPI number plus 32. Add 32 to the interrupt numbers from the peripherals to form the interrupt number that the GIC sees. GIC interrupts 0-31 are for internal use.

Table 5-4 CS3 region peripheral memory map (continued)

Peripheral	Modeled	Address range	Size	GIC Int ^q
Timer-1 (SP804)	Yes	0x00_1C120000-0x00_1C12FFFF	64KB	35
Virtio block device	Yes	0x00_1C130000-0x00_1C13FFFF	64KB	74
Virtio P9 device	Yes	0x00_1C140000-0x00_1C14FFFF	64KB	75
Reserved	-	0x00_1C130000-0x00_1C15FFFF	192KB	-
TwoWire serial interface (DVI) - unused	No	0x00_1C160000-0x00_1C16FFFF	64KB	-
Real-time Clock (PL031)	Yes	0x00_1C170000-0x00_1C17FFFF	64KB	36
Reserved	-	0x00_1C180000-0x00_1C19FFFF	128KB	-
CF Card - unused	No	0x00_1C1A0000-0x00_1C1AFFFF	64KB	
Reserved	-	0x00_1C1B0000-0x00_1C1EFFFF	256KB	-
Color LCD Controller (PL111)	Yes	0x00_1C1F0000-0x00_1C1FFFFF	64KB	46
Reserved	-	0x00_1C200000-0x00_1FFFFFFF	64KB	-

5.2.2 VE memory map for Cortex®-R series

The Versatile Express RS1 memory map with the RS2 extensions is the base of the global memory map for the Cortex-R series platform model.

Table 5-5 Cortex-R series VE FVP memory map

Memory	Modeled	Address range
DRAM	Yes	0x00000000-0x3FFFFFFF
FLASH0	Yes	0x40000000-0x43FFFFFF
FLASH1	Yes	0x44000000-0x47FFFFFF
PSRAM	Yes	0x48000000-0x4BFFFFFF
RAM	No	0x4C000000-0x4FFFFFFF
PL390 GIC CPU Interface [†]	Yes	0xAE000000-0xAE00FFFF
PL390 GIC Distributor [†]	Yes	0xAE001000-0xAE001FFF
VE System Registers	Yes	0xB0000000-0xB000FFFF
SP810	Yes	0xB0010000-0xB001FFFF
PL041 AACI	Yes	0xB0040000-0xB004FFFF
PL180 MCI	Yes	0xB0050000-0xB005FFFF
PL050 KMI0	Yes	0xB0060000-0xB006FFFF
PL050 KMI1	Yes	0xB0070000-0xB007FFFF
PL011 UART0	Yes	0xB0090000-0xB009FFFF
PL011 UART1	Yes	0xB00A0000-0xB00AFFFF
PL011 UART2	Yes	0xB00B0000-0xB00BFFFF
PL011 UART3	Yes	0xB00C0000-0xB00CFFFF

[†] Cortex-R4 and Cortex-R5 models only.

Table 5-5 Cortex-R series VE FVP memory map (continued)

Memory	Modeled	Address range
SP805 WATCHDOG	Yes	0xB00F0000–0xB00FFFFF
TIMER_0_1	Yes	0xB0110000–0xB011FFFF
TIMER_2_3	Yes	0xB0120000–0xB012FFFF
PL031 Real Time Clock	Yes	0xB0170000–0xB017FFFF
Compact Flash	No	0xB01A0000–0xB01AFFFF
PL011 UART4	Yes	0xB01B0000–0xB01BFFFF
PL111 CLCD	Yes	0xB01F0000–0xB01FFFFF ^S
RAM	No	0xB4000000–0xBBFFFFFF
Video RAM	Yes	0xBC000000–0xBDFFFFFF
Ethernet (SMSC 91C111)	Yes	0xBE000000–0xBEFFFFFF
USB	No	0xBF000000–0xBFFFFFFF

^S For Cortex-R4 and Cortex-R5 models, the range is 0xA0000000–0xA0010000.

5.3 Interrupt maps for VE FVPs

This section describes the interrupt maps for the VE FVPs.

This section contains the following subsections:

- [5.3.1 VE - interrupt assignments for Cortex®-A series on page 5-68.](#)
- [5.3.2 VE - interrupt assignments for Cortex®-R series on page 5-68.](#)

5.3.1 VE - interrupt assignments for Cortex®-A series

The platform routes the following *Shared Peripheral Interrupts* (SPIs) to the GIC.

Table 5-6 SPI GIC assignments

IRQ ID	SPI offset	Device
32	0	Watchdog, SP805
34	2	Dual timer 0/1, SP804
35	3	Dual timer 2/3, SP804
36	4	Real-time Clock, PL031
37	5	UART0, PL011
38	6	UART1, PL011
39	7	UART2, PL011
40	8	UART3, PL011
41	9	MCI, PL180, MCIINTR0
42	10	MCI, PL180, MCIINTR1
43	11	AACI, PL041
44	12	KMI - Keyboard, PL050
45	13	KMI - Mouse, PL050
46	14	Color LCD Controller, PL111
47	15	Ethernet, SMSC 91C111
74	42	Virtio block device
75	43	Virtio P9 device
92	60	CPU 0 PMU
93	61	CPU 1 PMU
94	62	CPU 2 PMU
95	63	CPU 3 PMU
117	85	HDLCD

5.3.2 VE - interrupt assignments for Cortex®-R series

This section describes the interrupt assignments.

Table 5-7 Interrupt assignments for Cortex-R4 and Cortex-R5

GIC IRQ ID	SPI offset	Device
32	0	Watchdog, SP805
34	2	Dual timer 0/1, SP804
35	3	Dual timer 2/3, SP804
36	4	Real-time Clock, PL031
41	9	MCI, PL180, MCIINTR0
42	10	MCI, PL180, MCIINTR1
43	11	AACI, PL041
44	12	KMI - Keyboard, PL050
45	13	KMI - Mouse, PL050
47	15	Ethernet, SMSC 91C111
75	16	Level 2 Cache Controller, PL310 Combined interrupt
76	14	Color LCD Controller, PL111
96	5	UART0, PL011
97	6	UART1, PL011
98	7	UART2, PL011
99	8	UART3, PL011

Table 5-8 Interrupt assignments for Cortex-R7 and Cortex-R8

GIC IRQ ID	SPI offset	Device
32	0	Watchdog, SP805
34	2	Dual timer 0/1, SP804
35	3	Dual timer 2/3, SP804
36	4	Real-time Clock, PL031
37	5	UART0, PL011
38	6	UART1, PL011
39	7	UART2, PL011
40	8	UART3, PL011
41	9	MCI, PL180, MCIINTR0
42	10	MCI, PL180, MCIINTR1
43	11	AACI, PL041
44	12	KMI - Keyboard, PL050
45	13	KMI - Mouse, PL050
46	14	Color LCD Controller, PL111
47	15	Ethernet, SMSC 91C111

5.4 VE parameters

This section describes the VE FVP instantiation parameters.

This section contains the following subsections:

- [5.4.1 VE instantiation parameters on page 5-70.](#)
- [5.4.2 VE secure memory parameters on page 5-70.](#)
- [5.4.3 VE switch S6 on page 5-70.](#)

5.4.1 VE instantiation parameters

This section describes the instantiation parameters for VE models.

Table 5-9 VE instantiation parameters

Component	Parameter	Type	Allowed values	Default value	Description
ve_sysregs	user_switches_value	Integer	See 5.4.3 VE switch S6 on page 5-70.	0	Switch S6 setting.
flashloader0	fname	String	Valid filename	""	Path to flash image file.
flashloader1	fname	String	Valid filename	""	Path to flash image file.
mmc	p_mmc_file	String	Valid filename	mmc.dat	Multimedia card filename.
pl111_clcd	pixel_double_limit	Integer	-	0x12c	Sets threshold in horizontal pixels below which pixels sent to framebuffer are doubled in size in both dimensions.
sp810_sysctrl	use_s8	Boolean	true, false	false	Indicates whether to read boot_switches_value.

5.4.2 VE secure memory parameters

This section describes the VE FVP secure memory parameters that you can change when you start the model.

Table 5-10 VE secure memory parameters

Name	Type	Allowed values	Default value	Description
daughterboard.secure_memory	Boolean	true, false	false	<p>false</p> <p>The platform behaves as before.</p> <p>true</p> <p>The address space is segregated according to the security mode of the core. Some memory blocks near the bottom of the address space are available to Secure transactions only. The rest of the address space is available to Non-secure transactions only.</p>

5.4.3 VE switch S6

This section describes the behavior and default positions of the VE system model switch.

Switch S6 is equivalent to the Boot Monitor configuration switch on the VE hardware.

If you have the standard Arm Boot Monitor flash image loaded, the setting of switch S6-1 changes what happens on model reset. Otherwise, the function of switch S6 is implementation dependent.

To write the switch position directly to the S6 parameter in the model, you must convert the switch settings to an integer value from the equivalent binary, where 1 is on and 0 is off.

Table 5-11 Default positions of VE system model switch

Switch	Default position	Function in default position
S6-1	OFF	Displays prompt permitting Boot Monitor command entry after system start.
S6-2	OFF	See STDIO redirection, below.
S6-3	OFF	See STDIO redirection, below.
S6-4 to S6-8	OFF	Reserved for application use.

If S6-1 is in the ON position, the Boot Monitor executes the boot script that was loaded into flash. If there is no script, the Boot Monitor prompt is displayed.

The settings of S6-2 and S6-3 affect STDIO source and destination on model reset.

Table 5-12 STDIO redirection

S6-2	S6-3	Output	Input	Description
OFF	OFF	UART0	UART0	STDIO autodetects whether to use semihosting I/O or a UART. If a debugger is connected, STDIO is redirected to the debugger output window, otherwise STDIO goes to UART0.
OFF	ON	UART0	UART0	STDIO is redirected to UART0, regardless of semihosting settings.
ON	OFF	CLCD	Keyboard	STDIO is redirected to the CLCD and keyboard, regardless of semihosting settings.
ON	ON	CLCD	UART0	STDIO output is redirected to the LCD and input is redirected to UART0, regardless of semihosting settings.

5.5 VE - components

A complete model implementation of the VE platform includes both VE-specific components and generic components, such as buses and timers.

To see a list of all the component instances in the model, run it with the `--list-instances` option.

The generic components are documented in the Fast Models Reference Manual, see [Core components](#) and [Fast Models components](#).

This section contains the following subsections:

- [5.5.1 VEVisualisation component on page 5-72](#).
- [5.5.2 VE_SysRegs component on page 5-75](#).

5.5.1 VEVisualisation component

This section describes the VEVisualisation component.

VEVisualisation - about

This component can generate events from the host mouse or keyboard when the visualization window is in focus. For example, you can toggle the switch elements from the visualization window.

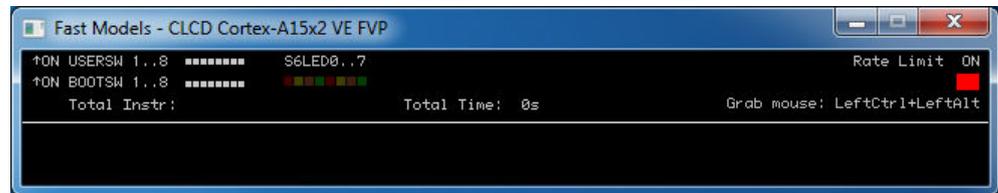


Figure 5-2 Startup VE FVP CLCD visualization window

When a suitable application or system image loads, and configures the PL111_CLCD controller registers, the window expands to show the contents of the frame buffer.

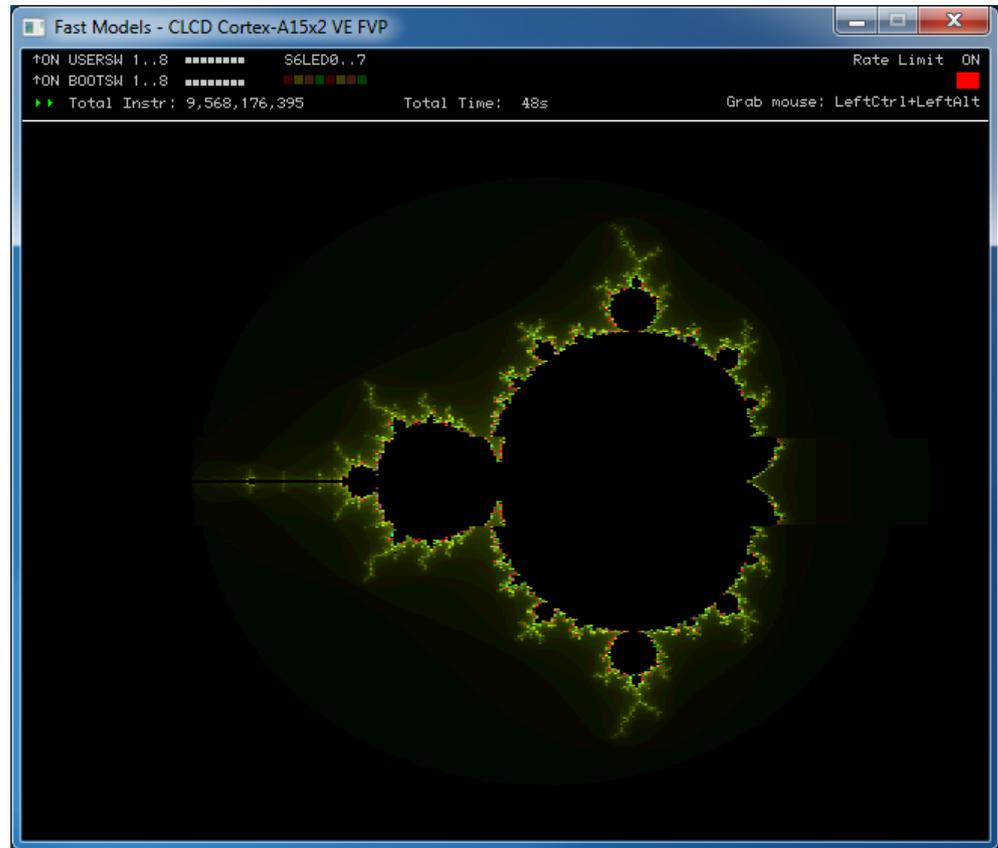


Figure 5-3 VE FVP CLCD with brot.axf image

The VEVisualisation LISA+ component can be found in the \$PVLIB_HOME/examples/LISA/FVP_VE/LISA/ directory.

Note

Using this component can reduce simulation performance. Use the `rate_limit-enable` parameter to control simulation speed.

VEVisualisation - ports

This section describes the VEVisualisation component ports.

Table 5-13 VEVisualisation ports

Name	Protocol	Type	Description
boot_switch	ValueState	Slave	Provides state for the eight Boot DIP switches on the right side of the CLCD status bar.
clock_50Hz	ClockSignal	Slave	50Hz clock input.
cluster0_ticks[4]	InstructionCount	Slave	Connection from processor model in cluster 0 to show its current instruction count.
cluster1_ticks[4]	InstructionCount	Slave	Connection from processor model in cluster 1 to show its current instruction count.
daughter_leds	ValueState	Slave	A read/write port to read and set the value of the LEDs. 1 bit per LED, LSB left-most, up to 32 LEDs available. The LEDs appear only when parameter <code>daughter_led_count</code> is set to nonzero.

Table 5-13 VEVisualisation ports (continued)

Name	Protocol	Type	Description
daughter_user_switches	ValueState	Slave	A read port to return the value of the daughter user switches. Write to this port to set the value of the switches, and use during reset only. LSB is left-most, up to 32 switches available.
keyboard	KeyboardStatus	Master	Output port providing key change events when the visualization window is in focus.
lcd	LCD	Slave	Connection from a CLCD controller for visualization of the frame buffer.
lcd_layout	LCDLayoutInfo	Master	Layout information for alphanumeric LCD display.
leds	ValueState	Slave	Displays state using the eight colored LEDs on the status bar.
mouse	MouseStatus	Master	Output port providing mouse movement and button events when the visualization window is in focus.
touch_screen	MouseStatus	Master	Provides mouse events when the visualization window is in focus.
user_switches	ValueState	Slave	Provides state for the eight User DIP switches on the left side of the CLCD status bar, equivalent to switch S6 on VE hardware.

VEVisualisation - parameters

This section describes the configuration parameters.

The syntax to use in a configuration file or on the command line is:

```
motherboard.vis.parameter=value
```

————— **Note** —————

Setting the `rate_limit-enable` parameter to `true` (the default) prevents the simulation from running too fast on fast workstations and enables timing loops and mouse actions to work correctly. However, it reduces the overall simulation speed. If your priority is high simulation speed, set `rate_limit-enable` to `false`.

Table 5-14 VEVisualisation parameters

Name	Type	Allowed values	Default value	Description
cluster0_name	string	-	Cluster0	Label for cluster 0 performance values.
cluster1_name	string	-	Cluster1	Label for cluster 1 performance values.
cpu_name	string	-	-	Processor name displayed in window title.
daughter_led_count	int	0-32	0	Set to nonzero to display up to 32 LEDs. See the <code>daughter_leds</code> port.
daughter_user_switch_count	int	0-32	0	Set this parameter to display up to 32 switches. See the <code>daughter_user_switches</code> port.
disable_visualisation	bool	true, false	false	Disable the VEVisualisation component on model startup.
rate_limit-enable	bool	true, false	true	Restrict simulation speed so that simulation time more closely matches real time rather than running as fast as possible.

Table 5-14 VEVisualisation parameters (continued)

Name	Type	Allowed values	Default value	Description
recorder.checkInstructionCount	bool	true, false	true	Check instruction count in recording file against actual instruction count during playback.
recorder.playbackFileName	string	-	""	Playback filename (empty string disables playback).
recorder.recordingFileName	string	-	""	Recording filename (empty string disables recording).
recorder.recordingTimeBase	int	-	0x5F5E100	Timebase in 1/s (relative to the master clock (where 100000000 means 10 nanoseconds resolution simulated time for a 1Hz master clock)) for recording (higher values give higher time resolution, playback timebase is always taken from the playback file).
recorder.verbose	int	-	0x0	Enable verbose messages (1=normal, 2=even more).
trap_key	int	Valid ATKeyCode key value	74, 0x4A ^t	Trap key that works with left Ctrl key to toggle mouse display.
window_title	string	-	"Fast Models - CLCD %cpu%"	Window title (cpu_name replaces %cpu%).

VEVisualisation - verification and testing

This component passes tests by use as an I/O device for booting Linux and other operating systems.

VEVisualisation - performance

Arm expects the elements in the status bar to have little effect on the performance of PV systems. However, applications that often redraw the contents of the frame buffer might incur overhead through GUI interactions on the host OS.

VEVisualisation - library dependencies

This component relies on the *Simple DirectMedia Layer* (SDL) libraries, specifically libSDL2-2.0.so.0.4.0.

This library is bundled with the Model Library and is also available as an rpm for Red Hat Enterprise Linux. On Windows, the library is called SDL2.d11.

5.5.2 VE_SysRegs component

This section describes the VE system registers component.

VE_SysRegs - about

This LISA+ component is a model of the VE status and system control registers.

VE_SysRegs - ports

This section describes the ports.

^t This is equivalent to the left **Alt** key, so pressing Left Alt and Left Ctrl simultaneously toggles the mouse display.

Table 5-15 VE_SysRegs ports

Name	Protocol	Type	Description
cb[0-1]	VECBProtocol	Master	The <i>Configuration Bus</i> (CB) controls the power and reset sequence.
clock_24Mhz	ClockSignal	Slave	Reference clock for internal counter register.
clock_100Hz	ClockSignal	Slave	Reference clock for internal counter register.
clock_CLCD	ClockRateControl	Master	The clock for the LCD controller.
lcd	LCD	Master	Multimedia bus interface output to the LCD.
leds	ValueState	Master	Displays state of the SYS_LED register using the eight colored LEDs on the status bar.
mmb[0-2]	LCD	Slave	Multimedia bus interface input.
mmc_card_present	StateSignal	Slave	Indicates the presence of a <i>MultiMedia Card</i> (MMC) image.
pvbus	PVBus	Slave	Slave port for connection to PV bus master/decoder.
system_reset	Signal	Master	Signal to the platform a complete system reset. Writes to the System Configuration registers can trigger the reset signal.
user_switches	ValueState	Master	Provides state for the eight User DIP switches on the left side of the CLCD status bar, equivalent to switch S6 on VE hardware.

VE_SysRegs - parameters

This section describes the parameters.

Table 5-16 VE_SysRegs parameters

Name	Type	Default value	Description
exit_on_shutdown	bool	false	Used to shut down the system. When true, if software uses the SYS_CFGCTRL function SYS_CFG_SHUTDOWN, then the simulator shuts down and exits. ^u
mmbSiteDefault	int	0x1	Default MultiMedia Bus (MMB) source (0=motherboard, 1=daughterboard 1, 2=daughterboard 2).
sys_proc_id0	int	0x0c000000	Processor ID register at CoreTile Express Site 1.
sys_proc_id1	int	0xff000000	Processor ID at CoreTile Express Site 2.
tilePresent	bool	true	Tile fitted.
user_switches_value	int	0	User switches.

VE_SysRegs - registers

This section describes the configuration registers.

Table 5-17 VE_SysRegs registers

Name	Offset	Access	Description
SYS_ID	0x00	Read/write	System identity.
SYS_SW	0x04	Read/write	Bits[7:0] map to switch S6.
SYS_LED	0x08	Read/write	Bits[7:0] map to user LEDs.

^u For more information on the SYS_CFGCTRL function values, see the *Motherboard Express μATX V2M-P1 Technical Reference Manual*.

Table 5-17 VE_SysRegs registers (continued)

Name	Offset	Access	Description
SYS_100HZ	0x24	Read only	100Hz counter.
SYS_FLAGS	0x30	Read/write	General purpose flags.
SYS_FLAGSCLR	0x34	Write only	Clear bits in general purpose flags.
SYS_NVFLAGS	0x38	Read/write	General purpose non-volatile flags.
SYS_NVFLAGSCLR	0x3C	Write only	Clear bits in general purpose non-volatile flags.
SYS_MCI	0x48	Read only	MCI.
SYS_FLASH	0x4C	Read/write	Flash control.
SYS_CFGSW	0x58	Read/write	Boot select switch.
SYS_24MHZ	0x5C	Read only	24MHz counter.
SYS_MISC	0x60	Read/write	Miscellaneous control flags.
SYS_DMA	0x64	Read/write	DMA peripheral map.
SYS_PROCID0	0x84	Read/write	Processor ID.
SYS_PROCID1	0x88	Read/write	Processor ID.
SYS_CFGDATA	0xA0	Read/write	Data to be read/written from/to motherboard controller.
SYS_CFGCTRL	0xA4	Read/write	Control data transfer to motherboard controller.
SYS_CFGSTAT	0xA8	Read/write	Status of data transfer to motherboard.

VE_SysRegs - verification and testing

This component was tested as part of the Versatile Express model.

5.6 Differences between the VE hardware and the system model

This section describes features of the hardware that the models do not implement, or implement with significant differences.

This section contains the following subsections:

- [5.6.1 Memory map on page 5-78.](#)
- [5.6.2 Memory aliasing on page 5-78.](#)
- [5.6.3 VE hardware features absent on page 5-78.](#)
- [5.6.4 VE hardware features different on page 5-78.](#)
- [5.6.5 Restrictions on the processor models on page 5-78.](#)
- [5.6.6 Timing considerations for the VE FVPs on page 5-79.](#)

5.6.1 Memory map

The model represents the memory map of the hardware VE platform, but is not an accurate representation of a specific revision.

The memory map in the supplied model is sufficiently complete and accurate to boot the same operating system images as for the VE hardware.

In the memory map, memory regions that are not explicitly occupied by a peripheral or by memory are unmapped. This includes regions otherwise occupied by a peripheral that is not implemented, and those areas that are documented as reserved. Accessing these regions from the host processor results in the model presenting a warning.

5.6.2 Memory aliasing

The model implements address-space aliasing of the DRAM. This means that the same physical memory locations are visible at different addresses.

The lower 2GB of the DRAM is accessible at `0x00_80000000`. The full 4GB of DRAM is accessible at `0x08_00000000` and again at `0x80_00000000`. The aliasing of DRAM then repeats from `0x81_00000000` up to `0xFF_FFFFFFFF`.

5.6.3 VE hardware features absent

These FVPs do not implement the following features of the hardware:

- Two-wire serial bus interfaces.
- USB interfaces.
- PCI Express interfaces.
- Compact flash.
- *Digital Visual Interface (DVI)*.
- Debug and test interfaces.
- *Dynamic Memory Controller (DMC)*.
- *Static Memory Controller (SMC)*.

5.6.4 VE hardware features different

These Fixed Virtual Platforms only partially implement some features of the hardware.

The partially implemented features might not work as you expect. Check the model release notes for the latest information.

Sound

The VE FVPs implement the PL041 AACI PrimeCell and the audio CODEC as in the VE hardware, but with a limited number of sample rates.

5.6.5 Restrictions on the processor models

Some general restrictions apply to the Fixed Virtual Platform implementations of Arm processors.

- The simulator does not model cycle timing. In aggregate, all instructions execute in one processor master clock cycle, except for Wait For Interrupt.
- Write buffers are not modeled on all processors.
- Most aspects of TLB behavior are implemented in the models. In Armv7 models and later, the TLB memory attribute settings are used when stateful cache is enabled.
- No device-accurate MicroTLB is implemented.
- A single memory access port is implemented. The port combines accesses for instruction, data, DMA, and peripherals. Configuration of the peripheral port memory map register is ignored.
- All memory accesses are atomic and are performed in *Programmer's View* (PV) order. Unaligned accesses are always performed as byte transfers.
- Some instruction sequences are executed atomically so that system time does not advance during their execution. This difference in behavior can affect sequential access of device registers where devices are expecting time to move on between each access.
- Interrupts are not taken at every instruction boundary.
- Integration and test registers are not implemented.
- Not all CP14 debug registers are implemented on all processors.
- Breakpoint types that the model supports directly are:
 - Single address unconditional instruction breakpoints.
 - Single address unconditional data breakpoints.
 - Unconditional instruction address range breakpoints.
- Pseudoregisters in the debugger support processor exception breakpoints. Setting an exception register to a nonzero value stops execution on entry to the associated exception vector.
- Performance counters are not implemented on all models.

5.6.6 Timing considerations for the VE FVPs

The Rate Limit feature matches simulation time to wall-clock time.

The *Fixed Virtual Platforms* (FVPs) provide an environment that enables running software applications in a functionally-accurate simulation. However, because of the relative balance of fast simulation speed over timing accuracy, there are situations where the models might behave unexpectedly.

When code interacts with real world devices like timers and keyboards, data arrives in the modeled device in real world, or wall clock, time, but simulation time can run much faster than the wall clock. This means that a single key press might register as several repeated key presses, or a single mouse click incorrectly becomes a double click.

Enabling Rate Limit, either using the Rate Limit button in the CLCD display, or the `rate_limit-enable` model instantiation parameter, forces the model to run at wall-clock time. This avoids issues with two clocks running at significantly different rates. For interactive applications, Arm recommends enabling Rate Limit.