Document History

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>31 August 2008</td>
<td>Non-Confidential</td>
<td>RealView Development Suite v4.0 Professional, System Generator v4.0 SP1.</td>
</tr>
<tr>
<td>B</td>
<td>31 December 2008</td>
<td>Non-Confidential</td>
<td>Fast Models v4.1. Added changes related to ARM_RTSM_PATH.</td>
</tr>
<tr>
<td>D</td>
<td>30 November 2009</td>
<td>Non-Confidential</td>
<td>RealView Development Suite v4.0 Professional edition.</td>
</tr>
<tr>
<td>F</td>
<td>30 September 2010</td>
<td>Non-Confidential</td>
<td>RealView Development Suite v4.1 SP1 Professional edition.</td>
</tr>
<tr>
<td>G</td>
<td>31 October 2012</td>
<td>Non-Confidential</td>
<td>DS-5 v5.12.</td>
</tr>
<tr>
<td>H</td>
<td>31 December 2012</td>
<td>Non-Confidential</td>
<td>DS-5 v5.13.</td>
</tr>
<tr>
<td>J</td>
<td>30 June 2015</td>
<td>Non-Confidential</td>
<td>DS-5 v5.15.</td>
</tr>
<tr>
<td>K</td>
<td>30 September 2013</td>
<td>Non-Confidential</td>
<td>DS-5 v5.16.</td>
</tr>
<tr>
<td>L</td>
<td>31 December 2013</td>
<td>Non-Confidential</td>
<td>DS-5 v5.17.</td>
</tr>
<tr>
<td>M</td>
<td>30 June 2014</td>
<td>Non-Confidential</td>
<td>DS-5 v5.19.</td>
</tr>
<tr>
<td>N</td>
<td>30 September 2014</td>
<td>Non-Confidential</td>
<td>DS-5 v5.20.</td>
</tr>
<tr>
<td>O</td>
<td>20 March 2015</td>
<td>Non-Confidential</td>
<td>DS-5 v5.21.</td>
</tr>
</tbody>
</table>

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

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU 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/about/trademark-usage-guidelines.php

Copyright © [2008-2010, 2012-2015], ARM Limited or its affiliates. All rights reserved.


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

ARM DS-5 EB FVP Reference Guide

Preface
  About this book ................................................................. 6
  Feedback ............................................................................. 7

Chapter 1  Introduction
  1.1 System model concepts .................................................. 1-9
  1.2 EB FVP concepts .......................................................... 1-10
  1.3 EB FVP composition ..................................................... 1-11

Chapter 2  Getting Started with EB FVPs
  2.1 Starting EB FVPs .......................................................... 2-15
  2.2 Configuring EB FVPs .................................................... 2-16
  2.3 EB FVP CLCD window .................................................. 2-17
  2.4 Using Ethernet with EB FVPs ......................................... 2-20
  2.5 Using a terminal with a system model ............................ 2-22
  2.6 Virtual File System ...................................................... 2-24
  2.7 Path names for using the VFS with pre-built FVPs ............... 2-26

Chapter 3  Programmer’s Reference for the EB FVPs
  3.1 EB model memory map .................................................. 3-28
  3.2 EB model configuration parameters ................................. 3-31
  3.3 Differences between the EB and CoreTile hardware and the models .......... 3-37
Preface

This preface introduces the ARM DS-5 EB FVP Reference Guide.

It contains the following:

About this book

This document describes how to configure and use the Emulation Baseboard Fixed Virtual Platforms. The models let you run software applications on a virtual implementation of an EB and an attached CoreTile. The EB FVPs are models of ARM application processors.

Using this book

This book is organized into the following chapters:

**Chapter 1 Introduction**
This chapter introduces the *Emulation Baseboard (EB) Fixed Virtual Platforms (FVPs)*.

**Chapter 2 Getting Started with EB FVPs**
This chapter introduces the procedures for starting and configuring EB FVPs, and running a software application on the model. The procedures differ depending on the ARM software tools you are using.

**Chapter 3 Programmer’s Reference for the EB FVPs**
This chapter introduces the memory map and the configuration registers for the peripheral and system component models.

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* *italic*
Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

*monospace* *bold*
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.
• The number ARM DUI0424O.
• 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.

——— END ———
Chapter 1
Introduction

This chapter introduces the *Emulation Baseboard (EB) Fixed Virtual Platforms* (FVPs).

It contains the following sections:

- *1.1 System model concepts* on page 1-9.
- *1.2 EB FVP concepts* on page 1-10.
- *1.3 EB FVP composition* on page 1-11.
1.1 System model concepts

This section describes the Fixed Virtual Platforms (FVPs) and the Programmer’s View (PV) models. FVPs enable development of software without the requirement for actual hardware.

The software provides PV models of processors and devices. The functional behavior of a model is equivalent to real hardware.

PV models sacrifice absolute timing accuracy to achieve fast simulated execution speed. You can rely on the PV models for confirming software functionality, but not for the accuracy of cycle counts, low-level component interactions, or other hardware-specific behavior.
1.2 EB FVP concepts

This section describes the Emulation Baseboard (EB), CoreTiles, and EB Fixed Virtual Platforms (FVPs).

The EB and CoreTiles are hardware development platforms that ARM® produces.

The EB FVPs are software models of the systems. ARM develops them using the Fast Models software.

Note

ARM provides the EB FVPs as example platform implementations. They are not accurate representations of a specific EB hardware revision. The FVPs support selected peripherals as this document describes. The supplied FVPs are sufficiently complete and accurate to boot the same operating system images as the EB hardware.

Related references

1.3 EB FVP composition on page 1-11.
1.3 EB FVP composition

This section describes the Fixed Virtual Platform components for the EB Reference System.

- Processor CoreTile:
  - Cortex®-A8.
- EB model with:
  - 64MB Flash memory.
  - 256MB RAM.
  - Ethernet interface.
  - UART interface.
  - Debug DIP switches and LEDs.
  - Real-Time Clock (RTC).
  - Time of year clock.
  - Programmable clock generators.
  - Synchronous Serial Port Interface (SSPI).
  - DMA controller configuration registers.
  - Static Memory Controller (SMC).

The EB FVPs also include virtual components:
- Visualization for Color LCD (CLCD) display, keyboard, and mouse.
- Touch screen controller.
- Four telnet terminals.

The FVPs for the EB Reference System are hierarchical models that consist of:

- Top-level view of the model.
- EB model.
- CoreTile model that the system model uses.

The EB FVPs provide a functionally accurate model for software execution. However, the model sacrifices timing accuracy to increase simulation speed. Key deviations from actual hardware are:

- Timing is approximate.
- Buses are simplified.
The CoreTile component provides the processor version and the associated ports that enable interconnection with other top-level components.

The block diagram of the EB Baseboard model shows the components in block form, the ports, and the interconnections.
Related references

3.2 EB model configuration parameters on page 3-31.
3.3 Differences between the EB and CoreTile hardware and the models on page 3-37.
Chapter 2
Getting Started with EB FVPs

This chapter introduces the procedures for starting and configuring EB FVPs, and running a software application on the model. The procedures differ depending on the ARM software tools you are using.

It contains the following sections:
• 2.1 Starting EB FVPs on page 2-15.
• 2.2 Configuring EB FVPs on page 2-16.
• 2.3 EB FVP CLCD window on page 2-17.
• 2.4 Using Ethernet with EB FVPs on page 2-20.
• 2.5 Using a terminal with a system model on page 2-22.
• 2.6 Virtual File System on page 2-24.
• 2.7 Path names for using the VFS with pre-built FVPs on page 2-26.
2.1 Starting EB FVPs

This section describes how to start EB FVPs.

To start an FVP, change to the directory where your model file is and enter at the command prompt:

```
model_name [--cadi-server] [--config-file filename] [-C instance.parameter=value] [-- application app_filename]
```

- `model_name` is the name of the model file. By default this file name is `FVP_EB_processor`.
- `--cadi-server` is an option that starts a CADI debug server allowing a CADI-enabled debugger, such as ARM DS-5 Debugger, to connect to the running model. You can configure it to wait for a debugger connection before starting.
- `filename` is the name of the optional plain text configuration file. Configuration files simplify managing multiple parameters.
- `instance.parameter=value` is the optional direct setting of a configuration parameter.
- `app_filename` is the file name of an image for your model to load at startup.

Starting the model opens the FVP CLCD display.

Related references

- 2.2 Configuring EB FVPs on page 2-16.
- 2.3 EB FVP CLCD window on page 2-17.

Related information

2.2 Configuring EB FVPs

This section describes how to configure the initial state of EB FVPs on the command line.

Using a configuration file

On the command line, use the --config-file option to start an FVP with a configuration file.

Begin comment lines in the optional plain text configuration file with a # character.

Each active line of the configuration file contains:

- Component instance name.
- Parameter name and the value.

Set boolean values with true/false or 1/0. Enclose strings in double quotes if they contain whitespace.

Typical configuration file

```plaintext
# Disable semihosting using true/false syntax
coretile.core.semihosting-enable=false
#
# Enable the boot switch using 1/0 syntax
baseboard.sp810_sysctrl.use_s8=1
#
# Set the boot switch position
baseboard.eb_sysregs_0.boot_switch_value=1
```

This file is equivalent to starting the model on the command line with:

```
-C coretile.core.semihosting-enable=false
-C baseboard.sp810_sysctrl.use_s8=1
-C baseboard.eb_sysregs_0.boot_switch_value=1
```

Using the command line

You can use the -C switch to define model parameters when you invoke the model (--parameter is a synonym for the -C switch). Use the same syntax as for a configuration file, but precede each parameter with the -C switch.

Booting a model from a flash image with boot options

Example for configuring an EB FVP

```
# Boot from a flash image
FVP_EB_Cortex-A8 \
  --parameter "baseboard.flashldr_0.fname=flash.bin" \
  --parameter "baseboard.eb_sysregs_0.user_switches_value=4" \
  --parameter "visualisation.disable_visualisation=false" \
  --parameter "visualisation.rate_limit-enable=0"
```

Related references

- 2.1 Starting EB FVPs on page 2-15.
- 3.2 EB model configuration parameters on page 3-31.
2.3 EB FVP CLCD window

This section describes the CLCD window for EB FVPs.

When FVPs start, the FVP CLCD window opens. This window represents the contents of the simulated color LCD frame buffer. It automatically resizes to match the horizontal and vertical resolution set in the CLCD peripheral registers.

![Figure 2-1 CLCD window at startup](image)

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

**USERSW**

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

These represent switch S6 on the EB hardware, USERSW[8:1], which is mapped to bits [7:0] of the SYS_SW register at address 0x10000004.

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

**BOOTSW**

Eight white boxes showing the state of the EB Boot DIP switches.

These represent switch S8 on the EB hardware, BOOTSEL[8:1], which is mapped to bits [15:8] of the SYS_SW register at address 0x100000004.

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 rather than by 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 EB User LEDs.

These represent LEDs D[21:14] on the EB hardware. These LEDs are mapped to bits [7:0] of the SYS_LED register at address 0x10000008. The boxes correspond to the red/yellow/green LEDs on the EB hardware.

**Total Instr**

A counter showing the total number of instructions executed.

The FVP models provide a programmer’s view of the system, so the CLCD displays total instructions rather than total core cycles. Timing can differ substantially from the hardware because:

- Bus fabric is simplified.
- Memory latencies are minimized.
- Programmer’s view core and peripheral models are used.

In general, the timing of operations within a 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 prevents the simulation from running faster than wall clock time.
Because the system model is highly optimized, your code can run faster than it would on real hardware. This difference can cause timing issues.
Rate Limit is enabled by default. Simulation time is restricted so that it more closely matches real time.
Click the square button to disable or enable Rate Limit. The text changes from ON to OFF and the colored box becomes darker when Rate Limit is disabled.

Note
You can control whether Rate Limit is enabled by using the rate_limit-enable parameter when instantiating the model.

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

Figure 2-2  CLCD window with Rate Limit off

Instr/sec
Shows the number of instructions that is executed per second of wall clock time.

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

You can reset the simulation counters by resetting the model.

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. This 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 on the right of the keyboard does not have the same effect.

If you prefer to use a different key, use the trap_key configuration option.

Related references
Switch S6 on page 3-32.
Switch S8 on page 3-32.
3.3.9 Timing considerations on page 3-40.
3.2.7 Visualization parameters on page 3-35.

Related information

2.4 Using Ethernet with EB FVPs

This section describes how to use Ethernet with EB FVPs.

This section contains the following subsections:

- 2.4.1 Host requirements on page 2-20.
- 2.4.2 Target requirements on page 2-20.
- 2.4.3 Configuring Ethernet on page 2-21.

2.4.1 Host requirements

To use the Ethernet capability of EB FVPs, you must set up your host computer.

Related information

Fast Models User Guide.

2.4.2 Target requirements

The EB FVPs provide you with a virtual Ethernet component.

SMSC91C111 Ethernet controller component

This model of the SMSC91C111 Ethernet controller uses a TAP device to communicate with the network.

Configure a HostBridge component to perform read and write operations on the TAP device. The HostBridge component is a virtual programmer’s view 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.

By default, the Ethernet component is disabled. Your target OS must include a driver for this device. Configure the kernel to use the SMSC chip. Linux supports the SMSC91C111 enabled

The default state of the enabled parameter is false. When the device is disabled, the kernel cannot detect the device.

mac_address

mac_address has two options. If you do not specify a MAC address, then when the simulator runs it takes the default MAC address for the Ethernet device. This address is randomly generated to increase the chance of it being unique when running models on multiple hosts on a local network.

Figure 2-3 Block diagram of the model networking structure
promiscuous
The default state is true, which means that it receives all network traffic, even any not addressed to the device. Use the default if you are using a single network device for multiple MAC addresses. Use it if, for example, you are sharing a network card between your host OS and the EB FVP Ethernet component.

Related references
3.2.3 Ethernet parameters on page 3-33.

Related information

2.4.3 Configuring Ethernet
You can configure a connection to the Ethernet interface on FVPs from Microsoft Windows or Linux.

Related information
Fast Models User Guide.
2.5 Using a terminal with a system model

This section describes how to use the Terminal component. This virtual component enables UART data transfer between a TCP/IP socket on the host and a serial port on the target.

This section contains the following subsections:
- 2.5.1 Terminal component in a system model on page 2-22.
- 2.5.2 Installing Telnet on Microsoft Windows 7 on page 2-23.

2.5.1 Terminal component in a system model

The TelnetTerminal block is what you configure when you define Terminal component parameters. The Virtual Machine is your EB FVP.

On the target side, the console process that your target OS invokes relies on the presence of a suitable driver. 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, EB FVPs start four telnet terminals when the model is initialized. To change the startup behavior for each of the four terminals, modify the corresponding component parameters.

If the terminal connection is broken, for example by closing a client telnet session, the port on the host opens. This port can 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.

You can define the port number of a terminal when an FVP starts. The actual value of the port that each terminal uses is declared when it starts or restarts. If that port is already in use, it cannot be the value that you specified. The port numbers are displayed in the host window in which you started the model.

You can start Terminal components in telnet mode or raw mode.
Telnet

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

Raw

Raw mode enables the byte stream to pass unmodified between the host and the target. In this state, the terminal 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.

2.5.2 Installing Telnet on Microsoft Windows 7

To use the Terminal component with a Microsoft Windows 7 client, 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 StartControl Panel Programs and Features. This action opens 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 action opens the Microsoft Windows Features dialog. Select the Telnet Client check box.
3. Click OK. The installation of Telnet can take several minutes to complete.
2.6 Virtual File System

The Virtual File System (VFS) allows your target to access parts of a host file system, through a target OS-specific driver and the MessageBox, a memory mapped device.

When using the VFS, access to the host file system is analogous to access to a shared network drive. You can expect it to behave in the same way.

VFS file system operations

- **getattr** retrieves metadata for the file, directory, or symbolic link.
- **mkdir** creates a directory.
- **remove** removes a file, directory, or symbolic link.
- **rename** renames a file, directory, or symbolic link.
- **rmdir** removes an empty directory.
- **setattr** sets metadata for the file, directory, or symbolic link.

**Note**

**setattr** is not implemented.

The VFS does not support symbolic links. The model cannot create hard links, but hard links that the host operating system creates function correctly.

VFS mount points

- **closemounts** frees an iterator handle that **openmounts** returned.
- **openmounts** retrieves an iterator handle for the list of available mounts.
- **readmounts** reads one entry from the mount iterator ID.

The VFS supports these directory iterators:

- **closedir** frees a directory iterator handle that **opendir** retrieved.
- **opendir** retrieves an iterator handle for the directory specified.
- **readdir** reads the next entry from the directory iterator.

**Note**

Datestamps are in milliseconds since the VFS epoch of January 01 1970 00:00 UTC; they are host datestamps.

VFS file operations

- **closefile** frees a handle that **openfile** opened.
- **filesync** forces the host OS to flush all file data to persistent storage.
getfilesize
returns the size of a file, in bytes.

openfile
returns a handle to the file.

readfile
reads a block of data from a file.

setfilesize
sets the size of a file in bytes, either by truncating, or extending the file with zeroes.

writefile
writes a block of data to a file.

Related references
2.7 Path names for using the VFS with pre-built FVPs on page 2-26.

Related information
WritingADriver.txt, %PVLIB_HOME%/VFS/docs/
2.7 Path names for using the VFS with pre-built FVPs

To use the VFS functionality of EB FVPs, set the `baseboard.vfs2.mount` configuration parameter when you start the model.

Set the parameter to hold the path to the host file system directory that the model is to access. All path names must be fully qualified paths:

`mountpoint:/path/to/object`

The supplied EB FVPs include the necessary VFS components.

Related references

2.6 Virtual File System on page 2-24.

Related information

This chapter introduces the memory map and the configuration registers for the peripheral and system component models.

It contains the following sections:

• 3.1 EB model memory map on page 3-28.
• 3.2 EB model configuration parameters on page 3-31.
• 3.3 Differences between the EB and CoreTile hardware and the models on page 3-37.
3.1 EB model memory map

This section describes the locations and interrupts of the memory, peripherals, and controllers that the EB Fixed Virtual Platforms use.

The EB FVP implementation of memory does not require you to program the memory controller with the correct values. To avoid application failure on hardware, be sure to set up the memory controller properly.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Modeled</th>
<th>Address range</th>
<th>Bus</th>
<th>Size</th>
<th>GIC Int&lt;sup&gt;a&lt;/sup&gt;</th>
<th>DCCI Int&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic memory</td>
<td>Yes</td>
<td>0x00000000–0xFFFFFFFF</td>
<td>AHB</td>
<td>256MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>System registers</td>
<td>Yes</td>
<td>0x10000000–0x10000000</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SP810 System Controller</td>
<td>Yes</td>
<td>0x10001000–0x10011FFF</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Two-Wire Serial Bus Interface</td>
<td>No</td>
<td>0x10002000–0x10021FFF</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x10003000–0x10031FFF</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PL041 Advanced Audio CODEC Interface (AACI)</td>
<td>Partial</td>
<td>0x10004000–0x1004FFFF</td>
<td>APB</td>
<td>4KB</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>PL180 MultiMedia Card Interface (MCI)</td>
<td>Partial&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0x10005000–0x1005FFFF</td>
<td>APB</td>
<td>4KB</td>
<td>49, 50</td>
<td>49, 50</td>
</tr>
<tr>
<td>Keyboard/Mouse Interface 0</td>
<td>Yes</td>
<td>0x10006000–0x10061FFF</td>
<td>APB</td>
<td>4KB</td>
<td>52</td>
<td>7</td>
</tr>
<tr>
<td>Keyboard/Mouse Interface 1</td>
<td>Yes</td>
<td>0x10007000–0x10071FFF</td>
<td>APB</td>
<td>4KB</td>
<td>53</td>
<td>8</td>
</tr>
<tr>
<td>Character LCD Interface</td>
<td>No</td>
<td>0x10008000–0x10081FFF</td>
<td>APB</td>
<td>4KB</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>UART 0 Interface</td>
<td>Yes</td>
<td>0x10009000–0x10091FFF</td>
<td>APB</td>
<td>4KB</td>
<td>44</td>
<td>4</td>
</tr>
<tr>
<td>UART 1 Interface</td>
<td>Yes</td>
<td>0x1000A000–0x100A1FFF</td>
<td>APB</td>
<td>4KB</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>UART 2 Interface</td>
<td>Yes</td>
<td>0x1000B000–0x100B1FFF</td>
<td>APB</td>
<td>4KB</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>UART 3 Interface</td>
<td>Yes</td>
<td>0x1000C000–0x100C1FFF</td>
<td>APB</td>
<td>4KB</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>Synchronous Serial Port Interface</td>
<td>Yes</td>
<td>0x1000D000–0x100D1FFF</td>
<td>APB</td>
<td>4KB</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>Smart Card Interface</td>
<td>No</td>
<td>0x1000E000–0x100E1FFF</td>
<td>APB</td>
<td>4KB</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x1000F000–0x100F1FFF</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SP805 Watchdog Interface</td>
<td>Yes</td>
<td>0x10010000–0x10011FFF</td>
<td>APB</td>
<td>4KB</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>SP804 Timer modules 0 and 1 interface (Timer 1 starts at 0x10011020)</td>
<td>Yes</td>
<td>0x10011000–0x100111FFF</td>
<td>APB</td>
<td>4KB</td>
<td>36</td>
<td>1</td>
</tr>
<tr>
<td>SP804 Timer modules 2 and 3 interface (Timer 3 starts at 0x10012020)</td>
<td>Yes</td>
<td>0x10012000–0x100121FFF</td>
<td>APB</td>
<td>4KB</td>
<td>37</td>
<td>2</td>
</tr>
<tr>
<td>PL061 GPIO Interface 0</td>
<td>Yes</td>
<td>0x10013000–0x100131FFF</td>
<td>APB</td>
<td>4KB</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>PL061 GPIO Interface 1</td>
<td>Yes</td>
<td>0x10014000–0x100141FFF</td>
<td>APB</td>
<td>4KB</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>PL061 GPIO Interface 2 (miscellaneous on board I/O)</td>
<td>Yes</td>
<td>0x10015000–0x100151FFF</td>
<td>APB</td>
<td>4KB</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x10016000–0x100161FFF</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<sup>a</sup> Add 32 to interrupt numbers from the peripherals (SPIs) to form the interrupt numbers that the GIC sees. GIC interrupts 0-31 are for internal use.

<sup>b</sup> Interrupt numbers that the DCCI system uses.

<sup>c</sup> The implementation of the PL180 is limited, so not all features are present.
Table 3-1  Memory map and interrupts for standard peripherals (continued)

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Modeled</th>
<th>Address range</th>
<th>Bus</th>
<th>Size</th>
<th>GIC Int&lt;sup&gt;a&lt;/sup&gt;</th>
<th>DCCI Int&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL030 Real-Time Clock Interface</td>
<td>Yes</td>
<td>0x10017000–0x10017FFF</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dynamic Memory Controller configuration</td>
<td>Partial</td>
<td>0x10018000–0x10018FFF</td>
<td>APB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PCI controller configuration registers</td>
<td>No</td>
<td>0x10019000–0x10019FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x1001A000–0x1001FFF</td>
<td>APB</td>
<td>24KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PL111 Color LCD Controller</td>
<td>Yes</td>
<td>0x10020000–0x1002FFFF</td>
<td>AHB</td>
<td>64KB</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>DMA Controller configuration registers</td>
<td>Yes</td>
<td>0x10030000–0x1003FFFF</td>
<td>AHB</td>
<td>64KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 1 (CPU interface)</td>
<td>Yes&lt;sup&gt;d&lt;/sup&gt;</td>
<td>0x10040000–0x10040FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 1 (Distributor interface)</td>
<td>Yes</td>
<td>0x10041000–0x10041FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 2 [CPU interface (nFIQ for tile 1)]</td>
<td>No&lt;sup&gt;d&lt;/sup&gt;</td>
<td>0x10050000–0x10050FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 2 (Distributor interface)</td>
<td>Yes</td>
<td>0x10051000–0x10051FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 3 [CPU interface (nIRQ for tile 2)]</td>
<td>No&lt;sup&gt;d&lt;/sup&gt;</td>
<td>0x10060000–0x10060FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 3 (Distributor interface)</td>
<td>No</td>
<td>0x10061000–0x10061FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 4 [CPU interface (nFIQ for tile 2)]</td>
<td>No&lt;sup&gt;d&lt;/sup&gt;</td>
<td>0x10070000–0x10070FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Generic Interrupt Controller 4 (Distributor interface)</td>
<td>No</td>
<td>0x10071000–0x10071FFF</td>
<td>AHB</td>
<td>4KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PL350 Static Memory Controller configuration&lt;sup&gt;e&lt;/sup&gt;</td>
<td>Yes</td>
<td>0x10080000–0x1008FFFF</td>
<td>AHB</td>
<td>64KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reserve</td>
<td>-</td>
<td>0x10090000–0x100EFFFF</td>
<td>AHB</td>
<td>448MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Debug Access Port (DAP) ROM table&lt;sup&gt;f&lt;/sup&gt;</td>
<td>No</td>
<td>0x100F0000–0x100FFFFF</td>
<td>AHB</td>
<td>64KB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x10100000–0x1FFFFFFF</td>
<td>-</td>
<td>255MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x20000000–0x3FFFFFFF</td>
<td>-</td>
<td>512MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NOR Flash</td>
<td>Yes&lt;sup&gt;g&lt;/sup&gt;</td>
<td>0x40000000–0x43FFFFFFF</td>
<td>AXI</td>
<td>64MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Disk on Chip</td>
<td>No</td>
<td>0x44000000–0x47FFFFFFF</td>
<td>AXI</td>
<td>64MB</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>SRAM</td>
<td>Yes</td>
<td>0x48000000–0x4bffFFFFF</td>
<td>AXI</td>
<td>64MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Configuration flash</td>
<td>No</td>
<td>0x4C000000–0x4dffFFFFF</td>
<td>AXI</td>
<td>32MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ethernet</td>
<td>Yes&lt;sup&gt;h&lt;/sup&gt;</td>
<td>0x4E000000–0x4effFFFFF</td>
<td>AXI</td>
<td>16MB</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>USB</td>
<td>No</td>
<td>0x4f000000–0x4fffffff</td>
<td>AXI</td>
<td>16MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PISMO expansion memory</td>
<td>No</td>
<td>0x50000000–0x5ffffefff</td>
<td>AXI</td>
<td>256MB</td>
<td>58</td>
<td>58</td>
</tr>
</tbody>
</table>

---

<sup>a</sup> Add 32 to interrupt numbers from the peripherals (SPIs) to form the interrupt numbers that the GIC sees. GIC interrupts 0-31 are for internal use.

<sup>b</sup> Interrupt numbers that the DCCI system uses.

<sup>c</sup> The EB FVP GICs differ from the EB hardware GICs because the register map is different.

<sup>d</sup> Although the EB hardware uses the PL093 static memory controller, the model implements PL350. These components are functionally equivalent.

<sup>e</sup> Some debuggers read information on the target processor and the debug chain from the DAP table.

<sup>f</sup> In EB FVPs, the IntelStrataFlashJ3 component implements this peripheral.

<sup>g</sup> In EB FVPs, the SC91C111 component implements this peripheral.
Table 3-1 Memory map and interrupts for standard peripherals (continued)

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Modeled</th>
<th>Address range</th>
<th>Bus</th>
<th>Size</th>
<th>GIC Int(^a)</th>
<th>DCCI Int(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI interface bus windows</td>
<td>No</td>
<td>0x60000000-0x6FFFFFFF</td>
<td>AXI</td>
<td>256MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dynamic memory (mirror)</td>
<td>Yes</td>
<td>0x70000000-0x7FFFFFFF</td>
<td>AXI</td>
<td>256MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Memory tile (Second CoreTile)</td>
<td>Yes</td>
<td>0x70000000-0x7FFFFFFF</td>
<td>AXI</td>
<td>0GB to 1GB</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Related references

3.3.2 Features partially implemented in the baseboard model on page 3-37.
3.3.6 Status and system control registers on page 3-39.
3.3.7 Generic Interrupt Controller on page 3-39.

Related information


\(^a\) Add 32 to interrupt numbers from the peripherals (SPIs) to form the interrupt numbers that the GIC sees. GIC interrupts 0-31 are for internal use.
\(^b\) Interrupt numbers that the DCCI system uses.
3.2 EB model configuration parameters

This section describes the EB model configuration parameters.

This section contains the following subsections:

- 3.2.1 About the EB model configuration parameters on page 3-31.
- 3.2.2 EB FVP baseboard parameters on page 3-31.
- 3.2.3 Ethernet parameters on page 3-33.
- 3.2.4 System controller parameters on page 3-33.
- 3.2.5 UART parameters on page 3-34.
- 3.2.6 Terminal parameters on page 3-34.
- 3.2.7 Visualization parameters on page 3-35.
- 3.2.8 FVP_EB_Cortex-A8 CoreTile parameters on page 3-36.

3.2.1 About the EB model configuration parameters

This section describes parameters that you can define at runtime. It does not describe parameters that you can modify only at build time, or that you would not normally modify in the hardware.

Related references

3.1 EB model memory map on page 3-28.
3.3 Differences between the EB and CoreTile hardware and the models on page 3-37.

3.2.2 EB FVP baseboard parameters

The baseboard has instantiation time parameters that you can change when you start the model.

The syntax to use in a configuration file is:

```
baseboard.component_name.parameter=value
```

Table 3-2 EB baseboard model instantiation parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameter</th>
<th>Type</th>
<th>Allowed values</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eb_sysregs_0</td>
<td>user_switches_value</td>
<td>int</td>
<td>-</td>
<td>0</td>
<td>Switch S6 setting.</td>
</tr>
<tr>
<td>eb_sysregs_0</td>
<td>boot_switch_value</td>
<td>int</td>
<td>-</td>
<td>0</td>
<td>Switch S8 setting.</td>
</tr>
<tr>
<td>flashldr_0</td>
<td>fname</td>
<td>string</td>
<td>Valid path</td>
<td>&quot;&quot;</td>
<td>Path to flash image file.</td>
</tr>
<tr>
<td></td>
<td>fnameWrite</td>
<td>string</td>
<td>Valid file name</td>
<td>&quot;&quot;</td>
<td>Name of flash image file.</td>
</tr>
<tr>
<td>flashldr_1</td>
<td>fname</td>
<td>string</td>
<td>Valid file name</td>
<td>&quot;&quot;</td>
<td>Path to flash image file.</td>
</tr>
<tr>
<td></td>
<td>fnameWrite</td>
<td>string</td>
<td>Valid file name</td>
<td>&quot;&quot;</td>
<td>Name of flash image file.</td>
</tr>
<tr>
<td>mmc</td>
<td>p_mmc_file</td>
<td>string</td>
<td>Valid file name</td>
<td>mmc.dat</td>
<td>Multimedia card file name.</td>
</tr>
<tr>
<td>pl111_clcd_0</td>
<td>pixel_double_limit</td>
<td>int</td>
<td>-</td>
<td>0x12c</td>
<td>Threshold in horizontal pixels below which pixels sent to framebuffer are doubled in size in both dimensions.</td>
</tr>
<tr>
<td>sdram_size</td>
<td>sdram_size</td>
<td>int</td>
<td>0 to 0x40000000</td>
<td>0</td>
<td>Size of the SDRAM on the second installed CoreTile.</td>
</tr>
<tr>
<td>sp805_watchdog_0</td>
<td>simhalt</td>
<td>bool</td>
<td>true,false</td>
<td>false</td>
<td>Enables or disables the ARM Watchdog Module (SP805).</td>
</tr>
</tbody>
</table>
Table 3-2  EB baseboard model instantiation parameters (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameter</th>
<th>Type</th>
<th>Allowed values</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp810_sysctrl</td>
<td>use_s8</td>
<td>bool</td>
<td>true,false</td>
<td>false</td>
<td>Whether to read boot_switches_value.</td>
</tr>
<tr>
<td>vfs2</td>
<td>mount</td>
<td>string</td>
<td>Valid file name</td>
<td>&quot;&quot;</td>
<td>Name of mount directory.</td>
</tr>
</tbody>
</table>

Switch S6

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

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

To write the switch position to the S6 parameter, convert the switch settings from binary to the equivalent integer value, where 1 is ON and 0 is OFF.

Table 3-3  Default positions for EB system model switch S6

<table>
<thead>
<tr>
<th>Switch</th>
<th>Default position</th>
<th>Function in default position</th>
</tr>
</thead>
<tbody>
<tr>
<td>S6-1</td>
<td>OFF</td>
<td>Displays prompt allowing Boot Monitor command entry after system start.</td>
</tr>
<tr>
<td>S6-2</td>
<td>OFF</td>
<td>See STDIO redirection.</td>
</tr>
<tr>
<td>S6-3</td>
<td>OFF</td>
<td>See STDIO redirection.</td>
</tr>
<tr>
<td>S6-4 to S6-8</td>
<td>OFF</td>
<td>Reserved for application use.</td>
</tr>
</tbody>
</table>

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 appears.

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

Table 3-4  STDIO redirection

<table>
<thead>
<tr>
<th>S6-2</th>
<th>S6-3</th>
<th>Output</th>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>UART0</td>
<td>UART0</td>
<td>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.</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>UART0</td>
<td>UART0</td>
<td>STDIO is redirected to UART0, regardless of semihosting settings.</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>CLCD</td>
<td>Keyboard</td>
<td>STDIO is redirected to the CLCD and keyboard, regardless of semihosting settings.</td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
<td>CLCD</td>
<td>UART0</td>
<td>STDIO output is redirected to the LCD and input is redirected to the keyboard, regardless of semihosting settings.</td>
</tr>
</tbody>
</table>

Related information

*Boot Monitor configuration and commands, Emulation Baseboard User Guide (Lead Free)*.

Switch S8

Switch S8 is disabled by default. To enable it, change the state of the parameter baseboard.sp810_sysctrl.use_s8 to true before you start the model.

If you have loaded a Boot Monitor flash image, switch S8 enables you to remap boot memory.

On reset, the EB hardware starts to execute code at 0x0, which is typically volatile DRAM. You can put the contents of nonvolatile RAM at this location by setting the S8 switch in the EB FVP CLCD. The settings take effect on model reset.
Table 3-5  EB system model switch S8 settings

<table>
<thead>
<tr>
<th>Switch S8[4:1]</th>
<th>Memory range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0x40000000-0x43FFFFFF</td>
<td>NOR flash (flash0, IntelStrataFlashJ3) remapped to 0x0.</td>
</tr>
<tr>
<td>0001</td>
<td>0x44000000-0x47FFFFFF</td>
<td>NOR flash (flash1, IntelStrataFlashJ3) remapped to 0x0.</td>
</tr>
<tr>
<td>0010</td>
<td>0x48000000-0x4BFFFFFF</td>
<td>SRAM (ssram, RAMDevice) remapped to 0x0.</td>
</tr>
</tbody>
</table>

Related references

3.2.2 EB FVP baseboard parameters on page 3-31.

3.2.3 Ethernet parameters

The Ethernet component has instantiation-time parameters that you can change.

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

baseboard.smsc_91c111_0.parameter=value

Table 3-6  Ethernet configuration parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Allowed values</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>bool</td>
<td>true,false</td>
<td>false</td>
<td>Host interface connection enabled.</td>
</tr>
<tr>
<td>mac_address</td>
<td>string</td>
<td>None,auto</td>
<td>00:02:f7:ef:31:11</td>
<td>Host/model MAC address.</td>
</tr>
<tr>
<td>promiscuous</td>
<td>bool</td>
<td>true,false</td>
<td>true</td>
<td>Put host into promiscuous mode, for example when sharing the Ethernet controller with the host OS.</td>
</tr>
</tbody>
</table>

mac_address

- If you do not specify a MAC address, then when you run the simulator, it takes the default MAC address and changes it. It replaces the bottom two bytes [00:02] with the bottom two bytes of the MAC address of one of the adaptors on the host workstation. This change increases the chance of the MAC address being unique when running models on multiple hosts on a local network.
- If you specify the MAC address as auto, the local MAC address is randomly generated each time that the simulator runs. The address has bit[1] set and bit[0] clear in the first byte to indicate a locally administered unicast MAC address.

Note

DHCP servers allocate IP addresses, but because they sometimes use the supplied MAC address, using random MAC addresses can conflict with them.

3.2.4 System controller parameters

The system controller has instantiation-time parameters that you can change when the model is started.

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

baseboard.sp810_sysctrl.parameter=value
### 3.2.5 UART parameters

The UART has instantiation-time parameters that you can change.

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

```plaintext
baseboard.uart_x.parameter=value
```

where \( x \) is the UART identifier 0, 1, 2, 3.

#### Table 3-8 UART configuration parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Allowed values</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>generic_uart</td>
<td>bool</td>
<td>true, false</td>
<td>false</td>
<td>Permit access only to the subset of registers defined as Generic Uart in the SBSA specification.</td>
</tr>
<tr>
<td>revision</td>
<td>string</td>
<td>-</td>
<td>&quot;r1p4&quot;</td>
<td>Revision to simulate (affects ID register and FIFO capacity).</td>
</tr>
<tr>
<td>shutdown_on_eot</td>
<td>bool</td>
<td>true, false</td>
<td>false</td>
<td>Shutdown simulation when an EOT (ASCII 4) char is transmitted.</td>
</tr>
<tr>
<td>shutdown_tag</td>
<td>string</td>
<td>-</td>
<td>&quot;&quot;</td>
<td>Shutdown simulation when a string is transmitted. Runtime parameter.</td>
</tr>
<tr>
<td>untimed_fifos</td>
<td>bool</td>
<td>true, false</td>
<td>true</td>
<td>Ignore the clock rate and transmit/receive serial data immediately.</td>
</tr>
</tbody>
</table>

### 3.2.6 Terminal parameters

The terminal has instantiation-time parameters that you can change when you start the model.

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

```plaintext
terminal_x.parameter=value
```

where \( x \) is the terminal ID 0, 1, 2, 3.

#### Table 3-9 Terminal instantiation parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Allowed values</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>string</td>
<td>telnet, raw</td>
<td>telnet</td>
<td>Terminal operation mode.</td>
</tr>
<tr>
<td>quiet</td>
<td>bool</td>
<td>true, false</td>
<td>false</td>
<td>Prevent output to stdout and stderr.</td>
</tr>
<tr>
<td>start_telnet</td>
<td>bool</td>
<td>true, false</td>
<td>false</td>
<td>Enable terminal when the system starts.</td>
</tr>
<tr>
<td>start_port</td>
<td>int</td>
<td>Valid port number</td>
<td>5000</td>
<td>Port used for the terminal when the system starts. If the specified port is not free, the port value is incremented by 1 until a free port is found.</td>
</tr>
</tbody>
</table>

\[1\] SYS_ID register value = 0x0225f500, corresponding to REV_A.
\[2\] SYS_ID register value = 0x12257500, corresponding to REV_B.
\[3\] SYS_ID register value = 0x22252500, corresponding to REV_C.
\[4\] For other values: SYS_ID register value = 0x0.
### Visualization parameters

The Visualization component has instantiation-time parameters that you can change when you start the model.

The syntax to use in a configuration file is:

```
visualisation.parameter=value
```

#### Table 3-10 Visualization instantiation parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Allowed values</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_name</td>
<td>string</td>
<td></td>
<td>&quot;&quot;</td>
<td>Name that appears in window title.</td>
</tr>
<tr>
<td>daughter_led_count</td>
<td>int</td>
<td>0-32</td>
<td>0</td>
<td>Number of LEDs that the daughter board has.</td>
</tr>
<tr>
<td>daughter_user_switch_count</td>
<td>int</td>
<td>0-32</td>
<td>0</td>
<td>Number of switches that the daughter board has.</td>
</tr>
<tr>
<td>disable_visualisation</td>
<td>bool</td>
<td>true,false</td>
<td>false</td>
<td>Disable the EBVisualisation component on model startup.</td>
</tr>
<tr>
<td>rate_limit-enable</td>
<td>bool</td>
<td>true,false</td>
<td>true</td>
<td>Restrict simulation speed so that simulation time more closely matches real time rather than running as fast as possible.</td>
</tr>
<tr>
<td>trap_key</td>
<td>int</td>
<td>Valid ATKeyCode key value</td>
<td>74^o</td>
<td>Trap key that works with Left Ctrl key to toggle mouse pointer display.</td>
</tr>
<tr>
<td>window_title</td>
<td>string</td>
<td>&quot;Fast Models - CLCD %cpu%&quot;</td>
<td></td>
<td>Window title (cpu_name replaces %cpu%).</td>
</tr>
</tbody>
</table>

#### Related references

2.3 EB FVP CLCD window on page 2-17.

#### Related information


---

^m You can click the Rate Limit button in the CLCD instead of setting the parameter at instantiation time.

^n If you have Fast Models installed, see the header file, %PVLIB_HOME%/components/KeyCode.h, for a list of ATKeyCode values. On Linux, see the file $PVLIB_HOME/components/KeyCode.h.

^o This is equivalent to the Left Alt key.
3.2.8 FVP_EB_Cortex-A8 CoreTile parameters

The Cortex-A8 CoreTile FVP has parameters that you can change when you start the model.

All listed parameters are instantiation-time parameters. This CoreTile FVP is based on r2p1 of the Cortex-A8 processor.

The syntax to use in a configuration file is:

`coretile.core.parameter=value`

The Cortex-A8 CoreTile FVP also includes a GIC but you cannot configure it at instantiation time.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Allowed values</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>semihosting-cmd_line</td>
<td>string</td>
<td>No limit except memory.</td>
<td>&quot;&quot;</td>
<td>Command line available to semihosting SVC calls.</td>
</tr>
<tr>
<td>semihosting-debug^P</td>
<td>bool</td>
<td>true,false</td>
<td>false</td>
<td>Enable debug output of semihosting SVC calls.</td>
</tr>
<tr>
<td>semihosting-enable</td>
<td>bool</td>
<td>true,false</td>
<td>true</td>
<td>Enable semihosting SVC traps.</td>
</tr>
<tr>
<td>semihosting-ARM_SVC</td>
<td>int</td>
<td>24-bit integer</td>
<td>0x123456</td>
<td>ARM SVC number for semihosting.</td>
</tr>
<tr>
<td>semihosting-Thumb_SVC</td>
<td>int</td>
<td>8-bit integer</td>
<td>0xAB</td>
<td>Thumb SVC number for semihosting.</td>
</tr>
<tr>
<td>semihosting-heap_base</td>
<td>int</td>
<td>0x00000000-0xFFFFFFFF</td>
<td>0x0</td>
<td>Virtual address of heap base.</td>
</tr>
<tr>
<td>semihosting-heap_limit</td>
<td>int</td>
<td>0x00000000-0xFFFFFFFF</td>
<td>0x0F000000</td>
<td>Virtual address of top of heap.</td>
</tr>
<tr>
<td>semihosting-stack_base</td>
<td>int</td>
<td>0x00000000-0xFFFFFFFF</td>
<td>0x10000000</td>
<td>Virtual address of base of descending stack.</td>
</tr>
<tr>
<td>semihosting-stack_limit</td>
<td>int</td>
<td>0x00000000-0xFFFFFFFF</td>
<td>0x0F000000</td>
<td>Virtual address of stack limit.</td>
</tr>
</tbody>
</table>

^P Ignored.
3.3 Differences between the EB and CoreTile hardware and the models

This section describes features of the Emulation Baseboard and CoreTile hardware that the models do not implement or implement differently.

This section contains the following subsections:
• 3.3.1 Features not present in the baseboard model on page 3-37.
• 3.3.2 Features partially implemented in the baseboard model on page 3-37.
• 3.3.3 Restrictions on the processor models on page 3-38.
• 3.3.4 Remapping and DRAM aliasing on page 3-39.
• 3.3.5 Dynamic memory characteristics on page 3-39.
• 3.3.6 Status and system control registers on page 3-39.
• 3.3.7 Generic Interrupt Controller on page 3-39.
• 3.3.8 GPIO2 on page 3-40.
• 3.3.9 Timing considerations on page 3-40.

3.3.1 Features not present in the baseboard model

The system models do not implement some features of the Emulation Baseboard hardware.

• Two wire serial bus interface.
• Character LCD interface.
• Smart card interface.
• PCI controller configuration registers.
• Debug access port.
• Disk on chip.
• Configuration flash.
• USB.
• PISMO expansion memory.
• PCI interface bus windows.
• UART Modem handshake signals.
• VGA support.

Related references
3.1 EB model memory map on page 3-28.

3.3.2 Features partially implemented in the baseboard model

The system models partially implement some features of the Emulation Baseboard hardware.

Partial implementation means that some of the components are present but the functionality has not been fully modeled. See the model release notes for the latest information.

Sound

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

DMC

The Dynamic Memory Controller (DMC), though modeled in the EB FVPs, does not provide direct memory access to all peripherals. Only the audio and synchronous serial port interface components can be accessed through the DMC.
3.3.3 Restrictions on the processor models

This section describes the restrictions on the processor models.

General restrictions on the processor models

Some general restrictions apply to the ARM processor FVP implementations.

• The Fast Models software does not model accurate instruction timing. A processor issues a set of instructions (a “quantum”) at a point in simulation time, and then waits before executing the next quantum. The processor averages one instruction per clock tick.
  — The perceived performance of software running on the models differs from real-world software. In particular, memory accesses and arithmetic operations all take the same amount of time.
  — A program can detect the quantized execution behavior of a processor, for example by polling a high-resolution timer.
  — All instructions in a quantum are effectively atomic. This atomicity can mask some race-condition bugs in software.

• Although the models include cache control registers, usually they only enable you to check register access permissions. Cache flush operations are supported, but they have no effect. As a consequence, code that can fail on real hardware because of cache aliasing problems can run without problems on EB FVPs.

• VFP and NEON™ instruction set execution on the models is not high performance.

• Write buffers are not modeled.

• The models implement most aspects of TLB behavior. In Architecture v7 models, the TLB memory attribute settings are used when stateful cache is enabled.

• No 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 order. All memory transactions are a maximum of 32 bits wide. Unaligned accesses are always performed as byte transfers.

• Interrupts are not taken at every instruction boundary.

• The semihosting-debug configuration parameter is ignored.

• Integration and test registers are not implemented.

• On some processor models, only one CP14 debug coprocessor register is included, CP14 DSCR. The register reads 0 and ignores writes. Access to other CP14 registers causes an undefined instruction exception. Use an external debugger to debug FVPs.

• The models directly support these breakpoint types:
  — 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.

• The models do not implement the Performance Management Unit (PMU) except for the instruction counter.

Related information


Restrictions on the FVP_EB_Cortex-A8 CoreTile

Additional restrictions apply to the Fixed Virtual Platform implementation of the Cortex-A8 processor.

• The model core sees two 4GB address spaces, one as seen from secure mode and one as seen from normal mode. The address spaces contain zero-wait state memory and peripherals, but much of the space is unmapped.
• The PLE model is purely register-based and has no implemented behavior.
• Unaligned accesses with the MMU disabled do not cause data aborts.

3.3.4 Remapping and DRAM aliasing

This section describes the differences between the hardware and the system model.

The EB hardware provides considerable memory remap functionality.

During this boot remapping, the bottom 64MB of the physical address map can be:
• NOR flash.
• Static expansion memory.

In addition to providing remap functionality, the hardware aliases all 256MB of system DRAM at 0x70000000.

Remapping does not typically apply to the system models. However, they do model NOR flash and can remap it.

In the memory map, the model does not map memory regions that peripherals or memory do not explicitly occupy. This simplification includes regions that unimplemented peripherals occupy in theory, and reserved areas. Accessing these regions from the host processor results in the model presenting a warning.

Related references

Switch S8 on page 3-32.

3.3.5 Dynamic memory characteristics

This section describes the differences between the hardware and the system model.

The Emulation Baseboard hardware contains a PL340 DMC. This component presents a configuration interface at address 0x10030000 in the memory map.

The system models configure a generic area of DRAM and do not model the PL340. This simplification helps speed the simulation.

3.3.6 Status and system control registers

This section describes the differences between the hardware and the system model.

For the hardware version of the Emulation Baseboard, the status and system control registers enable the processor to determine its environment and to control some on-board operations.

The system model implements all EB system registers except for SYS_TEST_OSC[4:0], the oscillator test registers. Unimplemented registers function as memory and the values that you write to them do not alter the behavior of the model.

The startup configuration of the EB FVP determines most of the functionality.

Related references

2.2 Configuring EB FVPs on page 2-16.

3.3.7 Generic Interrupt Controller

The Generic Interrupt Controller (GIC) that the EB FVPs provide differs substantially from the GIC in the Emulation Board firmware.

The programmer’s model of the newer device is largely backwards compatible. The model GIC is an implementation of the PL390 PrimeCell.

Related information

3.3.8 GPIO2

This section describes the differences between the hardware and the system model.

The system model provides the GPIO as a generic IO device. In the EB hardware, GPIO2 handles USB, a push button, and MCI status signals. The EB FVPs do not implement USB and MCI, and do not model a push button.

3.3.9 Timing considerations

This section describes the differences between the hardware and the system model.

The relative balance of fast simulation speed over timing accuracy means that there are situations where the models can behave unexpectedly.

The FVPs are environments that enable software applications to run in a functionally accurate simulation. However, if your code interacts with real world devices like timers and keyboards, timing differences can occur. The real world data arrives in the modeled device in real world (or wall clock) time, but simulation time can be running much faster than the wall clock. Therefore, a single keypress can appear to be several repeated key presses, or a single mouse click incorrectly becomes a double click.

The EB FVPs provide the Rate Limit feature to match simulation time to wall-clock time. Enabling Rate Limit, either by 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 feature avoids issues with two clocks running at different rates. For interactive applications, ARM recommends enabling Rate Limit.