virtutech

Simics/PM-PPC Target Guide

Simics Version	3.0
Revision	1406
Date	2008-02-20

VIRTUTECH CONFIDENTIAL

Contents

Chapter 1

About Simics Documentation

1.1 Conventions

Let us take a quick look at the conventions used throughout the Simics documentation. Scripts, screen dumps and code fragments are presented in a `monospace` font. In screen dumps, user input is always presented in bold font, as in:

```
Welcome to the Simics prompt
simics> this is something that you should type
```

Sometimes, artificial line breaks may be introduced to prevent the text from being too wide. When such a break occurs, it is indicated by a small arrow pointing down, showing that the interrupted text continues on the next line:

```
This is an artificial ↵
line break that shouldn't be there.
```

The directory where Simics is installed is referred to as `[simics]`, for example when mentioning the `[simics]/README` file. In the same way, the shortcut `[workspace]` is used to point at the user's workspace directory.

1.2 Simics Guides and Manuals

Simics comes with several guides and manuals, which will be briefly described here. All documentation can be found in `[simics]/doc` as Windows Help files (on Windows), HTML files (on Unix) and PDF files (on both platforms). The new Eclipse-based interface also includes Simics documentation in its own help system.

Simics Installation Guide for Unix and for Windows

These guides describe how to install Simics and provide a short description of an installed Simics package. They also cover the additional steps needed for certain features of Simics to work (connection to real network, building new Simics modules, . . .).

Simics User Guide for Unix and for Windows

These guides focus on getting a new user up to speed with Simics, providing information on Simics features such as debugging, profiling, networks, machine configuration and scripting.

Simics Eclipse User Guide

This is an alternative User Guide describing Simics and its new Eclipse-based graphical user interface.

Simics Target Guides

These guides provide more specific information on the different architectures simulated by Simics and the example machines that are provided. They explain how the machine configurations are built and how they can be changed, as well as how to install new operating systems. They also list potential limitations of the models.

Simics Programming Guide

This guide explains how to extend Simics by creating new devices and new commands. It gives a broad overview of how to work with modules and how to develop new classes and objects that fit in the Simics environment. It is only available when the DML add-on package has been installed.

DML Tutorial

This tutorial will give you a gentle and practical introduction to the Device Modeling Language (DML), guiding you through the creation of a simple device. It is only available when the DML add-on package has been installed.

DML Reference Manual

This manual provides a complete reference of DML used for developing new devices with Simics. It is only available when the DML add-on package has been installed.

Simics Reference Manual

This manual provides complete information on all commands, modules, classes and haps implemented by Simics as well as the functions and data types defined in the Simics API.

Simics Micro-Architectural Interface

This guide describes the cycle-accurate extensions of Simics (Micro-Architecture Interface or MAI) and provides information on how to write your own processor timing models. It is only available when the DML add-on package has been installed.

RELEASENOTES and LIMITATIONS files

These files are located in Simics's main directory (i.e., `[simics]`). They list limitations, changes and improvements on a per-version basis. They are the best source of information on new functionalities and specific bug fixes.

Simics Technical FAQ

This document is available on the Virtutech website at `http://www.simics.net/support`. It answers many questions that come up regularly on the support forums.

Simics Support Forum

The Simics Support Forum is the main support tool for Simics. You can access it at `http://www.simics.net`.

Other Interesting Documents

Simics uses Python as its main script language. A Python tutorial is available at `http://www.python.org/doc/2.4/tut/tut.html`. The complete Python documentation is located at `http://www.python.org/doc/2.4/`.

Chapter 2

Simics/PM-PPC Overview

Simics/PM-PPC models the *Artesyn PM/PPC* board. The PM/PPC board includes a PowerPC 750 processor, memory, flash memory, on-board ethernet interface, a CPC700 bridge with PCI support and 2 UARTS etc.

Linux and VxWorks are known to work on Simics/PM-PPC. Only files needed to boot Linux are availble from Virtutech.

Chapter 3

Simulated Machines

Simics scripts for starting PM-PPC machines are located in the `[workspace]/targets/pmppc/` directory, while the actual configuration scripts can be found in `[simics]/targets/pmppc/`.

3.1 Pmppc

The default configuration can be modified as described in section 3.2.

3.1.1 Pmppc Scripts

This chapter explains the files used to boot Linux on PM/PPC.

`orange-common.simics`
Starts the PM/PPC machine with the default configuration to boot linux.

3.2 Parameters for Machine Scripts

The following parameters can be set before running the `orange-common.simics` scripts. Other `*-linux-*.simics` scripts may set some of the parameters unconditionally, and do not allow the user to override them.

3.2.1 orange-common

$cpu_class
PowerPC processor to use (default ppc750)

$create_network
Set to `yes` if the script should create an Ethernet link and connect the primary Ethernet adapter to it.

$disk_size
Size of SCSI disk.

$disk_image
Initial contents of the of SCSI disk.

$do_boot
Set to `yes` if the script should automatically enter linux command line options and start linux.

$do_login
Set to `yes` if the script should automatically login as root on linux and configure the first ethernet device.

$eth_link
The Ethernet link to connect the primary Ethernet adapter to. This parameter should be set when a link already exist and the *$create_network* parameter is `no`.

$freq_mhz
The clock frequency in MHz for the processor.

$host_name
The host name used by the DHCP and DNS servers for this machine This variable will not change the host name set for the machine on the disk dumps.

$ip_address
The IP address used by the DHCP and DNS servers for this machine This variable will not change any IP address set for the machine on the disk dumps.

$kernel_image
The Linux kernel image file that should be booted.

$kernel_offset
The offset where the kernel image should be loaded.

$kernel_start
The start address for the kernel.

$mac_address
MAC address of the primary Ethernet adapter.

$tb_freq_mhz
The clock frequency of the time base on the processor.

$rtc_time
Date and time of the real-time clock at boot.

$service_node
The *service node* to use for DHCP and DNS. This parameter should be set when a service noce already exist and the *$create_network* parameter is `no`.

Chapter 4

Supported Components

The following sections list components that are supported for the PM-PPC architecture. There also exist other components in Simics, such as various PCI devices, that may work for PM-PPC but that have not been tested.

The default machines are constructed from components in the `-system.include` files in `[simics]/targets/pmppc/`. See the Configuration and Checkpointing chapter in the Simics User Guide for information on how to define your own machine, or make modifications to an existing machine.

4.1 PM/PPC Components

4.1.1 pmppc-board

Description

TODO:

Attributes

cpu_class

Required attribute; **read/write** access; type: **String**.

Processor type, PowerPC processor to use.

cpu_frequency

Required attribute; **read/write** access; type: **Float**.

Processor frequency in MHz.

mac_address

Required attribute; **read/write** access; type: **String**.

The MAC address of the Ethernet adapter.

rtc_time

Required attribute; **read/write** access; type: **String**.

The data and time of the Real-Time clock.

timebase_frequency
Required attribute; **read/write** access; type: **Float**.
Time-base frequency in MHz.

Commands

**create-pmppc-board [*"name"*] *"cpu_class" cpu_frequency timebase_frequency "mac_address" "rtc*

Creates a non-instantiated component of the class "pmppc-board". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-pmppc-board [*"name"*] *"cpu_class" cpu_frequency timebase_frequency "mac_address" "rtc_t*

Creates an instantiated component of the class "pmppc-board". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**pmppc-board**>**.info**
Print detailed information about the configuration of the device.

<**pmppc-board**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
eth0	ethernet-link	down
pci-slot2	pci-bus	down
pci-slot6	pci-bus	down
uart[0-1]	serial	down

4.2 PCI Device Components

4.2.1 pci-dec21143

Description
The "pci-dec21143" component represents a DEC21143 PCI based fast Ethernet adapter.

Attributes

bios
Optional attribute; **read/write** access; type: **String**.
The x86 BIOS file to use.

mac_address
Required attribute; **read/write** access; type: **String**.
The MAC address of the Ethernet adapter.

Commands

create-pci-dec21143 [*"name"*] *"mac_address"* [*"bios"*]
Creates a non-instantiated component of the class "pci-dec21143". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-dec21143>.info
Print detailed information about the configuration of the device.

<pci-dec21143>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

4.2.2 pci-sym53c875

Description
The "pci-sym53C875" component represents a SYM53C875PCI based SCSI controller.

Attributes

bios
Optional attribute; **read/write** access; type: **String**.
The x86 SCSI BIOS file to use.

Commands

create-pci-sym53c875 [*"name"*] [*"bios"*]
Creates a non-instantiated component of the class "pci-sym53c875". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<pci-sym53c875>.info
Print detailed information about the configuration of the device.

<pci-sym53c875>.status
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus	scsi-bus	down

4.3 Standard Components

4.3.1 ddr-memory-module

Description
The "ddr-memory-module" component represents a DDR memory module.

Attributes

banks
Optional attribute; **read/write** access; type: **Integer**.
Number of banks.

cas_latency
Optional attribute; **read/write** access; type: **Integer**.
CAS-latency; each set bit corresponds to a latency the memory can handle

columns
Optional attribute; **read/write** access; type: **Integer**.
Number of columns.

ecc_width
Optional attribute; **read/write** access; type: **Integer**.
The error correction width.

memory_megs
Pseudo attribute; **read-only** access; type: **Integer**.
Total about of memory in MB.

module_data_width
Optional attribute; **read/write** access; type: **Integer**.
The module SDRAM width.

module_type
Optional attribute; **read/write** access; type: **String**.
Type of memory.

primary_width
Optional attribute; **read/write** access; type: **Integer**.
Primary SDRAM width.

rank_density
Optional attribute; **read/write** access; type: **Integer**.
The rank density.

ranks
Optional attribute; **read/write** access; type: **Integer**.
Number of ranks (logical banks).

rows
Optional attribute; **read/write** access; type: **Integer**.
Number of rows.

speed
Optional attribute; **read/write** access; type: **String**.
PC standard speed. Supported values are PC2700 and none.

Commands

**create-ddr-memory-module [*"name"*] [*rows*] [*columns*] [*ranks*] [*module_data_width*] [*primary_w*

Creates a non-instantiated component of the class "ddr-memory-module". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

**new-ddr-memory-module [*"name"*] [*rows*] [*columns*] [*ranks*] [*module_data_width*] [*primary_wid*

Creates an instantiated component of the class "ddr-memory-module". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**ddr-memory-module**>**.info**
Print detailed information about the configuration of the device.

<**ddr-memory-module**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
mem-bus	mem-bus	up

4.3.2 std-ethernet-link

Description

The "std-ethernet-link" component represents a standard Ethernet link.

Attributes

frame_echo

Optional attribute; **read/write** access; type: **Integer**.

Set this attribute to echo frames back to the sender. Default is not to echo frames.

link_name

Optional attribute; **read/write** access; type: **String**.

The name to use for the **ethernet-link** object. An error will be raised at instantiation time if the link cannot be given this name.

Commands

create-std-ethernet-link [*"name"*] [*"link_name"*] [*frame_echo*]

Creates a non-instantiated component of the class "std-ethernet-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-ethernet-link [*"name"*] [*"link_name"*] [*frame_echo*]

Creates an instantiated component of the class "std-ethernet-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-ethernet-link**>**.info**

Print detailed information about the configuration of the device.

<**std-ethernet-link**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
device	ethernet-link	any

4.3.3 std-serial-link

Description

The "std-serial-link" component represents a standard Serial link.

Commands

create-std-serial-link [*"name"*]

Creates a non-instantiated component of the class "std-serial-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-serial-link [*"name"*]

Creates an instantiated component of the class "std-serial-link". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-serial-link**>**.info**

Print detailed information about the configuration of the device.

<**std-serial-link**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
serial[0-1]	serial	any

4.3.4 std-service-node

Description

The "std-service-node" component represents a network service node that can be connected to Ethernet links to provide services such as DNS, DHCP/BOOTP, RARP and TFTP. A service node component does not have any connectors by default. Instead, connectors have to be added using the <**std-service-node**>**.add-connector** command.

Attributes

dynamic_connectors

Optional attribute; **read/write** access; type: **[[iss]**∗].

List of user added connectors

next_connector_id

Optional attribute; **read/write** access; type: **Integer**.

Next service-node device ID.

Commands

create-std-service-node [*"name"*]
Creates a non-instantiated component of the class "std-service-node". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-service-node [*"name"*]
Creates an instantiated component of the class "std-service-node". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-service-node**>**.add-connector *"ip"* [*"netmask"*]**
Adds a connector to the service-node with specified IP address and netmask. A connector must be created for the service-node before an Ethernet link can be connected to it. The *ip* argument is the IP address that the service node will use on the link. The *netmask* argument is optional, and defaults to `255.255.255.0`. The name of the new connector is returned.

<**std-service-node**>**.info**
Print detailed information about the configuration of the device.

<**std-service-node**>**.status**
Print detailed information about the current status of the device.

4.3.5 std-scsi-bus

Description
The "std-scsi-bus" component represents a 16 slot SCSI bus.

Commands

create-std-scsi-bus [*"name"*]
Creates a non-instantiated component of the class "std-scsi-bus". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-scsi-bus**>**.info**
Print detailed information about the configuration of the device.

<**std-scsi-bus**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	any

4.3.6 std-scsi-disk

Description

The "std-scsi-disk" component represents a SCSI-2 disk.

Attributes

file

Optional attribute; **read/write** access; type: **String**.

File with disk contents for the full disk Either a raw file or a CRAFF file.

scsi_id

Required attribute; **read/write** access; type: **Integer**.

The ID on the SCSI bus.

size

Required attribute; **read/write** access; type: **Integer**.

The size of the SCSI disk in bytes.

Commands

create-std-scsi-disk [*"name"*] *scsi_id size* [*"file"*]

Creates a non-instantiated component of the class "std-scsi-disk". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<std-scsi-disk>.info

Print detailed information about the configuration of the device.

<std-scsi-disk>.status

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	up

4.3.7 std-scsi-cdrom

Description

The "std-scsi-cdrom" component represents a SCSI-2 CD-ROM.

Attributes

scsi_id

Required attribute; **read/write** access; type: **Integer**.

The ID on the SCSI bus.

Commands

create-std-scsi-cdrom [*"name"*] *scsi_id*

Creates a non-instantiated component of the class "std-scsi-cdrom". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-scsi-cdrom**>**.info**

Print detailed information about the configuration of the device.

<**std-scsi-cdrom**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	up

4.3.8 std-text-console

Description

The "std-text-console" component represents a serial text console.

Attributes

bg_color

Optional attribute; **read/write** access; type: **String**.

The background color.

fg_color

Optional attribute; **read/write** access; type: **String**.

The foreground color.

height
Optional attribute; **read/write** access; type: **Integer**.
The height of the console window.

title
Optional attribute; **read/write** access; type: **String**.
The Window title.

width
Optional attribute; **read/write** access; type: **Integer**.
The width of the console window.

win32_font
Optional attribute; **read/write** access; type: **String**.
Font to use in the console on Windows host.

x11_font
Optional attribute; **read/write** access; type: **String**.
Font to use in the console when using X11 (Linux/Solaris host).

Commands

**create-std-text-console [*"name"*] [*"title"*] [*"bg_color"*] [*"fg_color"*] [*"x11_font"*] [*"win32_font"*] [*w*

Creates a non-instantiated component of the class "std-text-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

**new-std-text-console [*"name"*] [*"title"*] [*"bg_color"*] [*"fg_color"*] [*"x11_font"*] [*"win32_font"*] [*wi*

Creates an instantiated component of the class "std-text-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-text-console**>**.info**
Print detailed information about the configuration of the device.

<**std-text-console**>**.status**
Print detailed information about the current status of the device.

Connectors

Name	**Type**	**Direction**
serial	serial	up

4.3.9 std-server-console

Description

The "std-server-console" component represents a serial console accessible from the host using telnet.

Attributes

telnet_port

Required attribute; **read/write** access; type: **Integer**.

TCP/IP port to connect the telnet service of the console to.

Commands

create-std-server-console [*"name"*] *telnet_port*

Creates a non-instantiated component of the class "std-server-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

new-std-server-console [*"name"*] *telnet_port*

Creates an instantiated component of the class "std-server-console". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

<**std-server-console**>**.info**

Print detailed information about the configuration of the device.

<**std-server-console**>**.status**

Print detailed information about the current status of the device.

Connectors

Name	Type	Direction
serial	serial	up

4.4 Base Components

The base components are abstract classes that contain generic component attributes and commands available for all components.

4.4.1 component

Description

Base component class, should not be instantiated.

Attributes

connections

Optional attribute; **read/write** access; type: **[[sos]**$*$**]**.

List of connections for the component. The format is a list of lists, each containing the name of the connector, the connected component, and the name of the connector on the other component.

connectors

Pseudo class attribute; **read-only** access; type: **D**.

Dictionary of dictionaries with connectors defined by this component class, indexed by name. Each connector contains the name of the connector "type", a "direction" ("up", "down" or "any"), a flag indicating if the connector can be "empty", another flag that is set if the connector is "hotplug" capable, and finally a flag that is TRUE if muliple connections to this connector is allowed.

instantiated

Optional attribute; **read/write** access; type: **b**.

Set to TRUE if the component has been instantiated.

object_list

Optional attribute; **read/write** access; type: **D**.

Dictionary with objects that the component consists of.

object_prefix

Optional attribute; **read/write** access; type: **String**.

Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: **Object**.

The top level component. Attribute is not valid until the component has been instantiated.

top_level

Optional attribute; **read/write** access; type: **b**.

Set to TRUE for top-level components, i.e. the root of a hierarchy.

4.4.2 top-component

Description

Base top-level component class, should not be instantiated.

Attributes

components

Optional attribute; **read/write** access; type: **[o∗]**.

List of components below the the top-level component. This attribute is not valid until the object has been instantiated.

cpu_list

Optional attribute; **read/write** access; type: **[o∗]**.

List of all processors below the the top-level component. This attribute is not valid until the object has been instantiated.

Chapter 5

Miscellaneous Notes

5.1 Changing the Processor Clock Frequency

The clock frequency of a simulated processor can be set arbitrarily in Simics. This will not affect the actual speed of simulation, but it will affect the number of instructions that need to be executed for a certain amount of simulated time to pass. If your execution only depends on executing a certain number of instructions, increasing the clock frequency will take the same amount of host time (but a shorter amount of target time). However, if there are time based delays of some kind in the simulation, these will take longer to execute.

At a simulated 1 MHz, one million target instructions will correspond to a simulated second (assuming the simple default timing of one cycle per instruction). At 100 MHz, on the other hand, it will take 100 million target instructions to complete a simulated second. So with a higher clock frequency, less simulated target time is going to pass for a certain period of host execution time.

If Simics is used to emulate an interactive system (especially one with a graphical user interface) it is a good idea to set the clock frequency quite low. Keyboard and mouse inputs events are handled by periodic interrupts in most operating systems, using a higher clock frequency will result in longer delays between invocations of periodic interrupts. Thus, the simulated system will feel slower in its user response, and update the mouse cursor position etc. less frequently. If this is a problem, the best technique for running experiments at a high clock frequency is to first complete the configuration of the machine using a low clock frequency. Save all configuration changes to a disk diff (like when installing operating systems). Then change the configuration to use a higher a clock frequency and reboot the target machine.

Note that for a lightly-loaded machine (for example, working at an interactive prompt on a serial console to an embedded Linux system), Simics will often execute quickly enough at the real target clock frequency that there is no need to artifically lower it.

5.2 Manually Testing Interrupts

Interrupts from the interrupt controller comes in to the Simics PowerPC via the `simple_interrupt` interface. To manually trigger an interrupt it is possible issue:

```
simics> @conf.cpu0.iface.simple_interrupt.interrupt(conf.cpu0, 0)
```

The command line triggers the interrupt towards the CPU. The seconds parameter (zero) indicates that this is a normal interrupt. Critical interrupts should use the value 1. The external interrupt will only be serviced (when continuing execution) if the MSR[EE] bit is set, enabling external interrupts. To manually set this bit issue:

```
simics> %msr = %msr | 1<<15
```

To lower the external interrupt manually issue:

```
simics> @conf.cpu0.iface.simple_interrupt.interrupt_clear(conf.cpu0, 0)
```

5.3 Cache Simulation

For generic information on how cache simulation is done in Simics please refer to Simics User Guide.

PowerPC instructions which manipulates the cache directly, such as `dcbf` can effect the cache model provided that the processor's `icache` and `dcache` are properly set.

The `icache` and `dcache` attributes should point to g-cache objects simulating instruction and data cache. For SMP configurations, the `cpu_group` attribute should point to a ppc-broadcast-bus object which will be informed about the caches the cpus uses. If the WIMG M-bit is set for a cache transaction, then memory coherency is required and the cache operation is sent down to the broadcast bus which distributes it to all known caches. For non-SMP configurations or if the M-bit is not set, the local cache is called directly.

The following operations are supported:

PowerPC operation	**Cache Operation**
dcbf (data cache block flush)	Cache_Control_Invalidate_Line
dcbst (data cache block store)	Cache_Control_Copyback_Line
dcbt (data cache block touch)	Cache_Control_Fetch_Line
dcbtst (data cache block touch for store)	Cache_Control_Fetch_Line
HID0[DCFI]	Cache_Control_Invalidate_Cache
HID0[ICFI]	Cache_Control_Invalidate_Cache

Other operations, such as locking cache lines, are not currently supported. The cache module receives the the cache operation via the cache_control interface.

Chapter 6

Limitations

Our model of the CPC700 and the PCI interrupt routing have some problems disallowing any PCI card to be connected at any port.

The linux support for the PM/PPC is not part of the standard linux kernel but have to be provided by Artesyn. This makes it more difficult to compile a kernel for the Simics model.

The subchapter below contains the limitations that exist on the relevant PowerPC processors that can be used with PM/PPC. The SPRs listed do currently have no associated side-effect when either the register is read or written. In many cases this is not a problem even when code do use these registers.

The unimplemented instructions will cause Simics to break execution if the are ever executed.

The instructions implemented as no-operation (NOPs) will just execute without any side-effects at all.

6.1 PowerPC 603e limitations

6.1.1 Unsupported SPRs

None

6.1.2 Miscellaneous Processor Core Limitations

PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Floating-point estimate instructions are not bit exact.

6.1.3 Unimplemented Instructions

eciwx
ecowx

6.2 PowerPC 750 limitations

6.2.1 Unsupported SPRs

SPR name	Number	Description
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
SIAR	955	Sampled Instruction Address

6.2.2 Miscellaneous Processor Core Limitations

PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.2.3 Unimplemented Instructions

eciwx
ecowx

6.3 PowerPC 750fx limitations

6.3.1 Unsupported SPRs

SPR name	Number	Description
HID2	1016	Hardware implementation register 2
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
SIAR	955	Sampled Instruction Address

6.3.2 Miscellaneous Processor Core Limitations

PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.3.3 Unimplemented Instructions

eciwx
ecowx

6.4 PowerPC 750gx limitations

6.4.1 Unsupported SPRs

SPR name	Number	Description
HID2	1016	Hardware implementation register 2
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
SIAR	955	Sampled Instruction Address

6.4.2 Miscellaneous Processor Core Limitations

PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.4.3 Unimplemented Instructions

eciwx
ecowx

6.5 PowerPC 755 limitations

6.5.1 Unsupported SPRs

SPR name	Number	Description
L2PM	1016	L2 Private Memory Control Register
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
SIAR	955	Sampled Instruction Address

6.5.2 Miscellaneous Processor Core Limitations

PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.5.3 Unimplemented Instructions

eciwx
ecowx

6.6 PowerPC 7400 limitations

6.6.1 Unsupported SPRs

SPR name	Number	Description
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
MMCR2	944	Monitor Control 2
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
SIAR	955	Sampled Instruction Address

6.6.2 Miscellaneous Processor Core Limitations

PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.6.3 Unimplemented Instructions

eciwx
ecowx

6.6.4 Instructions Implemented as NOPs

dss[all]

6.7 PowerPC 7447 limitations

6.7.1 Unsupported SPRs

SPR name	Number	Description
ICTRL	1011	Instruction cache/interrupt control register
LDSTCR	1016	Load/Store control register
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
MMCR2	944	Monitor Control 2
MSSSR0	1015	Memory subsystem status register
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
PMC5	945	Performance Counter 5
PMC6	946	Performance Counter 6
SIAR	955	Sampled Instruction Address

6.7.2 Miscellaneous Processor Core Limitations

MMU: 36 bit physical address `HID0[XAEN]` support is not implemented.
PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.7.3 Unimplemented Instructions

eciwx
ecowx

6.7.4 Instructions Implemented as NOPs

dss[all]

6.8 PowerPC 7450 limitations

6.8.1 Unsupported SPRs

SPR name	Number	Description
ICTRL	1011	Instruction cache/interrupt control register
L3PM	983	L3 private memory register
LDSTCR	1016	Load/Store control register
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
MMCR2	944	Monitor Control 2
MSSSR0	1015	Memory subsystem status register
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
PMC5	945	Performance Counter 5
PMC6	946	Performance Counter 6
SIAR	955	Sampled Instruction Address

6.8.2 Miscellaneous Processor Core Limitations

MMU: 36 bit physical address `HID0[XAEN]` support is not implemented.
PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.8.3 Unimplemented Instructions

eciwx
ecowx

6.8.4 Instructions Implemented as NOPs

dss[all]

6.9 PowerPC 7457 limitations

6.9.1 Unsupported SPRs

SPR name	Number	Description
ICTRL	1011	Instruction cache/interrupt control register
L3ITCR0	984	L3 cache input timing control register 0
L3ITCR1	1001	L3 cache input timing control register 1
L3ITCR2	1002	L3 cache input timing control register 2
L3ITCR3	1003	L3 cache input timing control register 3
L3OHCR	1000	L3 cache output hold control register
L3PM	983	L3 private memory register
LDSTCR	1016	Load/Store control register
MMCR0	952	Monitor Control 0
MMCR1	956	Monitor Control 1
MMCR2	944	Monitor Control 2
MSSSR0	1015	Memory subsystem status register
PMC1	953	Performance Counter 1
PMC2	954	Performance Counter 2
PMC3	957	Performance Counter 3
PMC4	958	Performance Counter 4
PMC5	945	Performance Counter 5
PMC6	946	Performance Counter 6
SIAR	955	Sampled Instruction Address

6.9.2 Miscellaneous Processor Core Limitations

MMU: 36 bit physical address `HID0[XAEN]` support is not implemented.
PMC: Performance Monitor Counters (PMC) are not supported.
Little endian mode setting in `MSR[LE]` is not implemented.
Setting ICTC[E] have no impact on instruction fetch cycles.
Floating-point estimate instructions are not bit exact.

6.9.3 Unimplemented Instructions

eciwx
ecowx

6.9.4 Instructions Implemented as NOPs

dss[all]

Index

S

T

Virtutech, Inc.

1740 Technology Dr., suite 460
San Jose, CA 95110
USA

Phone +1 408-392-9150
Fax +1 408-608-0430

http://www.virtutech.com