Imperas Guide to using Virtual Platforms

Platform / Module Specific Information for mips.ovpworld.org / MipsMalta

Imperas Software Limited

Imperas Buildings, North Weston
Thame, Oxfordshire, OX9 2HA, U.K.
docs@imperas.com.

Author	Imperas Software Limited
Version	20211118.0
Filename	Imperas_Platform_User_Guide_MipsMalta.pdf
Created	31 December 2021
Status	OVP Standard Release

Copyright Notice

Right to Copy Documentation

The license agreement with Imperas permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the readers responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED, AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Model Release Status

This model is released as part of OVP releases and is included in OVPworld packages. Please visit OVPworld.org.

Table Of Contents

1.0 Platform / Module: MipsMalta

This document provides the details of the usage of an Imperas OVP Virtual Platform / Module. The first half of the document covers specifics of this particular component. For more information about Imperas OVP virtual platforms, how they are built and used, please see the later sections in this document.

1.1 Virtual Platform / Module Type

Hardware described using OVP can either be a platform, module, processor, or peripheral.
This component has a purpose specified as being part of an Extendable Platform Kit (EPK). This is typically a platform that is part of a package that includes not only the platform but also software to run on the platform and scripts to control it.

1.2 Licensing

Open Source Apache 2.0

1.3 Description

This is a platform representing a MIPS Malta development board.
It provides the peripherals required to boot and run a Linux Operating System.
A single MIPS32 architecture processor is instantiated in this platform.
This instance could be duplicated to instantiate further processors to easily create an SMP platform.
Attributes are provided for configuration of the generic ISA model for a specific processor.
The processor model is configured to operate as a MIPS32 4KEc.

The main SDRAM and Flash memory is modeled using RAM models. Both are initialised in places by the
'SmartLoaderLinux'. The SmartLoaderLinux allows ease of use of changing kernel command lines,
loading an initial ram disk and creating the boot flash(s). The operation of the SmartloaderLinux is configured
using a number of attributes.
The kernel attribute of the SmartLoaderLinux and the imagefile of the processor must be consistent.

NOTE: a non Mips Malta peripheral 'AlphaDisplay16x2' has been defined in this platform definition
to be used for demo purposes. It should be removed if there is a memory error in the address space 0x18000100-0x18000103

If this platform is not part of your installation, then it is available for download from www.OVPworld.org/ip-vendor-mips.

1.4 Limitations

Verification has only been carried out using Little Endian memory ordering.

1.5 Reference

MIPS Malta User's Manual MD00048-2B-MALTA-USM-1.07.pdf
MIPS Malta-R Development Platform User's Manual MD00627-2B-MALTA_R-USM-01.01.pdf
CoreFPGA User's Manual MD00116-2B-COREFPGA-USM-01.00.pdf
Linux for the MIPS Malta Development Platform User's Guide MD00646-2B-LINUXMALTA-USM-01.03.pdf

1.6 Location

The MipsMalta virtual platform / module is located in an Imperas/OVP installation at the VLNV: mips.ovpworld.org / platform / MipsMalta / 1.0.

1.7 Platform Simulation Attributes

Table 1. Platform Simulation Attributes

Attribute	Value	Description
stoponctrlc	stoponctrlc	Stop on control-C

2.0 Command Line Control of the Platform

2.1 Built-in Arguments

Table 2. Platform Built-in Arguments

Attribute	Value	Description
allargs	allargs	The Command line parser will accept the complete imperas argument set. Note that this option is ignored in some Imperas products

When running a platform in a Windows or Linux shell several command arguments can be specified. Typically there is a '-help' command which lists the commands available in the platforms. For example:

```
myplatform.exe -help
```

Some command line arguments require a value to be provided. For example:

```
myplatform.exe -program myimagefile.elf
```

2.2 Platform Specific Command Line Arguments

Table 3. Platform Command Line Arguments

Name	Type	Description
kernel	stringvar	The Linux Kernel image e.g. vmlinux
ramdisk	stringvar	Boot Linux Kernel from the specified ramdisk image e.g. initrd.gz
disk	stringvar	Boot Linux Kernel from the root partition on the disk image e.g. mipsle_hda1
root	stringvar	Specify the root partition on the disk image e.g. /dev/sda1
pagebits	uns64var	Specify the page bits used by a Linux Kernel

console	stringvar	Specify the command line console entry of the Linux Kernel command line, for example tty0
finishonhalt	boolvar	Finish simulation when Malta Soft reset asserted
redir	boolvar	Enable re-direction of IP address
nographics	boolvar	Disable the VGA graphics window.
tftproot	stringvar	Enable TFTP and specify the tftp root on the host machine.
bootimage	stringvar	Specify a boot image to use. This replaces the default image generated by the SmartLoader.

3.0 Processor [mips.ovpworld.org/processor/mips32_r1r5/1.0] instance: mipsle1

3.1 Processor model type: 'mips32_r1r5' variant '24KEf' definition

Imperas OVP processor models support multiple variants and details of the variants implemented in this model can be found in:
- the Imperas installation located at ImperasLib/source/mips.ovpworld.org/processor/mips32_r1r5/1.0/doc
- the OVP website: OVP_Model_Specific_Information_mips32_r1r5_24KEf.pdf

3.1.1 Description

MIPS32 Configurable Processor Model

3.1.2 Licensing

Usage of binary model under license governing simulator usage. Source of model available under Imperas Software License Agreement.

3.1.3 Limitations

If this model is not part of your installation, then it is available for download from www.OVPworld.org/ip-vendor-mips.

3.1.4 Verification

Models have been validated correct as part of the MIPS Verified program and run through the MIPS AVP test programs

3.1.5 Features

only MIPS32 Instruction set implemented
MMU Type: Standard TLB
FPU implemented
L1 I and D cache model in either full or tag-only mode implemented (disabled by default)
Vectored interrupts implemented
MIPS16e ASE implemented
DSP ASE implemented

3.2 Instance Parameters

Several parameters can be specified when a processor is instanced in a platform. For this processor instance

'mipsle1' it has been instanced with the following parameters:

Table 4. Processor Instance 'mipsle1' Parameters (Configurations)

Parameter	Value	Description
endian	little	Select processor endian (big or little)
simulateexceptions	simulateexceptions	Causes the processor simulate exceptions instead of halting
mips	100.0	The nominal MIPS for the processor

Table 5. Processor Instance 'mipsle1' Parameters (Attributes)

Parameter Name	Value	Type
variant	24KEf	enum
vectoredinterrupt	0	bool
config1MMUSizeM1	63	uns32

3.3 Memory Map for processor 'mipsle1' bus: 'bus1'

Processor instance 'mipsle1' is connected to bus 'bus1' using master port 'INSTRUCTION'.
Processor instance 'mipsle1' is connected to bus 'bus1' using master port 'DATA'.

Table 6. Memory Map ('mipsle1' / 'bus1' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0xFFFFFFF	Core_Board_SDRAM	ram
0x10000000	0x1BFFFFFF	pciBr	bridge
0x1E000000	0x1E3FFFFF	map	bridge
0x1F000000	0x1F0008FF	maltaFpga	MaltaFPGA
0x1F000900	0x1F00093F	uartCBUS	16450C
0x1F000A00	0x1F000FFF	maltaFpga	MaltaFPGA
0x1FC00000	0x1FC0000F	remap1	bridge
0x1FC00010	0x1FC00013	Core_Board_SDRAM_promInit	SmartLoaderLinux
0x1FC00014	0x1FFFFFFF	remap2	bridge
0x20000000	0x5FFFFFFF	Core_Board_SDRAM2	ram
0x80000000	0xFFFFFFFF	high2low	bridge

Table 7. Bridged Memory Map ('mipsle1' / 'pciBr' / 'busPCI' [width: 32])

Lo Address	Hi Address	Instance	Component
remappable	remappable	PCI_NET	79C970
remappable	remappable	PCI_PM	PciPM
remappable	remappable	PIIX4-IDE	PciIDE
remappable	remappable	pciBrD	DynamicBridge
remappable	remappable	sysControl	GT6412x
remappable	remappable	vga	GD5446
0x100A0000	0x100BFFFF	vgaMemRegion	rom

Table 8. Bridged Memory Map ('mipsle1' / 'map' / 'flashBus' [width: 32])

Lo Address	Hi Address	Instance	Component
0x1E000000	0x1E3FFFFF	Monitor_Flash	ram

Table 9. Bridged Memory Map ('mipsle1' / 'remap1' / 'flashBus' [width: 32])

Lo Address	Hi Address	Instance	Component
0x1E000000	0x1E3FFFFF	Monitor_Flash	ram

Table 10. Bridged Memory Map ('mipsle1' / 'remap2' / 'flashBus' [width: 32])

Lo Address	Hi Address	Instance	Component
0x1E000000	0x1E3FFFFF	Monitor_Flash	ram

Table 11. Bridged Memory Map ('mipsle1' / 'high2low' / 'bus1' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0xFFFFFFF	Core_Board_SDRAM	ram
0x10000000	0x1BFFFFFF	pciBr	bridge
0x1E000000	0x1E3FFFFF	map	bridge
0x1F000000	0x1F0008FF	maltaFpga	MaltaFPGA
0x1F000900	0x1F00093F	uartCBUS	16450C
0x1F000A00	0x1F000FFF	maltaFpga	MaltaFPGA
0x1FC00000	0x1FC0000F	remap1	bridge
0x1FC00010	0x1FC00013	Core_Board_SDRAM_promInit	SmartLoaderLinux
0x1FC00014	0x1FFFFFFF	remap2	bridge
0x20000000	0x5FFFFFFF	Core_Board_SDRAM2	ram

3.4 Net Connections to processor: 'mipsle1'

Table 12. Processor Net Connections ('mipsle1')

Net Port	Net	Instance	Component
hwint0	i8259Int	intCtrlMaster	8259A

4.0 Peripheral Instances

4.1 Peripheral [ovpworld.org/peripheral/DynamicBridge/1.0] instance: pciBrD

4.1.1 Description

DynamicBridge - Dynamically enable/disable a bus bridge from the input slave port to the output master port. The bridge is enabled when the input net is high, disabled when it is low. The size of the port is defined with the portSize parameter. The address on the input slave port is defined by the spLoAddress parameter. The address on the output master port is defined by the mpLoAddress parameter. All three parameters must be specified. The input and output ports may be connected to the same bus.

4.1.2 Licensing

Open Source Apache 2.0

4.1.3 Limitations

The range of the input slave port must not conflict with any exiting port connected to the bus. The output bus width is hard coded to be 32 bits.

4.1.4 Reference

This is not based upon the operation of a real device

Table 13. Configuration options (attributes) set for instance 'pciBrD'

Attribute	Value	Type	Expression
spLoAddress	0x18000000	uns64	
mpLoAddress	0	uns64	
portSize	0x4ff	uns64	
enableBridge	1	bool	

4.2 Peripheral [mips.ovpworld.org/peripheral/SmartLoaderLinux/1.0] instance: Core_Board_SDRAM_promInit

4.2.1 Licensing

Open Source Apache 2.0

4.2.2 Description

Smart peripheral creates memory initialisation for a MIPS32 based Linux kernel boot. Performs the generation of boot code at the reset vector (virtual address 0xbfc00000) of the MIPS32 processor. Loads both the linux kernel and initial ramdisk into memory and patches the boot code to jump to the kernel start. Initialises the MIPS32 registers and Linux command line.

4.2.3 Reference

MIPS Malta User Manual. MIPS Boot code reference.

4.2.4 Limitations

None

There are no configuration options set for this peripheral instance.

4.3 Peripheral [marvell.ovpworld.org/peripheral/GT6412x/1.0] instance: sysControl

4.3.1 Licensing

Open Source Apache 2.0

4.3.2 Description

GT64120 System Controller.
Diagnostic levels:
PCI_SLAVE 0x03
PCI_CONFIG_MASTER 0x04

PCI_EMPTY 0x08
INT_ACK 0x10
MAIN_BUS 0x20
INFO 0x40

4.3.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.3.4 Reference
GT64121A System Controller for RC4650/4700/5000 and RM526X/527X/7000 CPUs Datasheet Revision 1.0 MAR 14, 2000

There are no configuration options set for this peripheral instance.

4.4 Peripheral [intel.ovpworld.org/peripheral/82371EB/1.0] instance: PIIX4

4.4.1 Licensing
Open Source Apache 2.0

4.4.2 Description
PIIX4 PCI configuration controller.

4.4.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.4.4 Reference
Intel 82371EB South Bridge Chipset Datasheet

Table 14. Configuration options (attributes) set for instance 'PIIX4'

Attribute	Value	Type	Expression
PCIslot	10	uns32	

4.5 Peripheral [intel.ovpworld.org/peripheral/PciIDE/1.0] instance: PIIX4-IDE

4.5.1 Licensing
Open Source Apache 2.0

4.5.2 Description
PCI:IDE interface. This forms part of the 82371 PIIX4 chip. It implements 4 IDE interfaces and 2 DMA controllers.

4.5.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.5.4 Reference
Intel 82371EB South Bridge Chipset Datasheet

Table 15. Configuration options (attributes) set for instance 'PIIX4-IDE'

Attribute	Value	Type	Expression
PCIslot	10	uns32	
PCIfunction	1	uns32	

4.6 Peripheral [intel.ovpworld.org/peripheral/PciPM/1.0] instance: PCI_PM

4.6.1 Licensing
Open Source Apache 2.0

4.6.2 Description
PCI Power Manager.

4.6.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.6.4 Reference
Intel 82371EB South Bridge Chipset Datasheet

Table 16. Configuration options (attributes) set for instance 'PCI_PM'

Attribute	Value	Type	Expression
PCIslot	10	uns32	
PCIfunction	3	uns32	

4.7 Peripheral [amd.ovpworld.org/peripheral/79C970/1.0] instance: PCI_NET

4.7.1 Licensing
Open Source Apache 2.0

4.7.2 Description
Implements part of the AMD AM79C97xx series Ethernet devices.
diagnosticlevel: bits 0:1 give levels for the network hardware. bits 2:3 give levels for the user:mode SLIRP interface.

4.7.3 Limitations
Sufficient is implemented to Boot MIPS Linux and support ethernet TCP/IP services.

4.7.4 Reference
AMD Am79C973/Am79C975 PCnet-FAST III Single-Chip 10/100 Mbps PCI Ethernet Controller with

Integrated PHY Datasheet

Table 17. Configuration options (attributes) set for instance 'PCI_NET'

Attribute	Value	Type	Expression
PCIslot	11	uns32	
PCIfunction	0	uns32	

4.8 Peripheral [intel.ovpworld.org/peripheral/8259A/1.0] instance: intCtrlMaster

4.8.1 Licensing
Open Source Apache 2.0

4.8.2 Description
Intel 8259A Programmable Interrupt Controller (PIT).

4.8.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.8.4 Reference
Intel 8259A Datasheet. MIPS Malta Platform Reference Guide.

Table 18. Configuration options (attributes) set for instance 'intCtrlMaster'

Attribute	Value	Type	Expression
spen	master	enum	

4.9 Peripheral [intel.ovpworld.org/peripheral/8259A/1.0] instance: intCtrlSlave

4.9.1 Licensing
Open Source Apache 2.0

4.9.2 Description
Intel 8259A Programmable Interrupt Controller (PIT).

4.9.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.9.4 Reference
Intel 8259A Datasheet. MIPS Malta Platform Reference Guide.

Table 19. Configuration options (attributes) set for instance 'intCtrlSlave'

Attribute	Value	Type	Expression

spen	slave	enum	

*4.10 Peripheral [ovpworld.org/peripheral/SerInt/1.0] instance: _SUPERIO_REG_*

4.10.1 Description
The serial interrupt control registers in the FDC 37M817 SuperIO device.

4.10.2 Limitations
This is a register description only. The model does not contain any functionality.

4.10.3 Licensing
Open Source Apache 2.0

4.10.4 Reference
SMsC FDC 37M817 SuperIO device datasheet

There are no configuration options set for this peripheral instance.

4.11 Peripheral [cirrus.ovpworld.org/peripheral/GD5446/1.0] instance: vga

4.11.1 Licensing
Open Source Apache 2.0

4.11.2 Description
Cirrus CL GD5446 VGA controller.

4.11.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform. The VGA peripheral utilises memory mapping. This requires the use of ICM memory for the frame buffers, which currently may stop its use in SystemC TLM2 platforms.

4.11.4 Reference
CL-GD5446 Preliminary Databook, Version 2.0, November 1996

Table 20. Configuration options (attributes) set for instance 'vga'

Attribute	Value	Type	Expression
scanDelay	50000	uns32	
PCIslot	18	uns32	
title	Imperas MIPS32 Malta	string	

4.12 Peripheral [intel.ovpworld.org/peripheral/Ps2Control/1.0] instance: Ps2Control

4.12.1 Licensing

Open Source Apache 2.0

4.12.2 Description
PS2 Keyboard/Mouse Controller.

4.12.3 Limitations
This is a preliminary model with sufficient functionality to enable Linux to Boot on the MIPS:MALTA platform. Mouse functions are currently turned off.

4.12.4 Reference
SMsC FDC37M817 Super I/O Controller Datasheet

Table 21. Configuration options (attributes) set for instance 'Ps2Control'

Attribute	Value	Type	Expression
pollPeriod	50000	uns32	
grabDisable	1	bool	

4.13 Peripheral [intel.ovpworld.org/peripheral/8253/1.0] instance: pit

4.13.1 Description
Intel 8253 Programmable Interval Timer (PIT)

4.13.2 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.
Not all modes are supported.

4.13.3 Licensing
Open Source Apache 2.0

4.13.4 Reference
Intel 8253 Datasheet. MIPS Malta Platform Reference Guide.

There are no configuration options set for this peripheral instance.

4.14 Peripheral [motorola.ovpworld.org/peripheral/MC146818/1.0] instance: rtc

4.14.1 Licensing
Open Source Apache 2.0

4.14.2 Description
MC146818 Real:time clock.

4.14.3 Limitations

This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.14.4 Reference
Motorola MC146818AS Datasheet

There are no configuration options set for this peripheral instance.

4.15 Peripheral [national.ovpworld.org/peripheral/16550/1.0] instance: uartTTY0

4.15.1 Licensing
Open Source Apache 2.0

4.15.2 Description
16550 UART model
The serial input/output from the simulator is implemented using the Serial Device Support described in OVP BHM and PPM API Functions Reference, which describes the parameters that control how the model interacts with the host computer.
Interrupts and FIFOs are supported.
Registers are aligned on 1 byte boundaries.

4.15.3 Limitations
Resolution of the baud rate is limited to the simulation time slice (aka quantum) size.
Values written to the MCR are ignored. Loopback mode is not supported.
The LSR is read-only. The model never sets the LSR 'Parity Error', 'Framing Error', 'Break Interrupt' or 'Error in RCVR FIFO' bits.
The MSR 'Data Set Ready' and 'Clear To Send' bits are set at reset and all other MSR bits are cleared. MSR bits will only be changed by writes to the MSR and values written to the Modem Status Register do not effect the operation of the model.

4.15.4 Reference
PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs datasheet (http://www.ti.com/lit/ds/symlink/pc16550d.pdf)

Table 22. Configuration options (attributes) set for instance 'uartTTY0'

Attribute	Value	Type	Expression
outfile	uartTTY0.log	string	
finishOnDisconnect	1	bool	

4.16 Peripheral [national.ovpworld.org/peripheral/16550/1.0] instance: uartTTY1

4.16.1 Licensing
Open Source Apache 2.0

4.16.2 Description

16550 UART model
The serial input/output from the simulator is implemented using the Serial Device Support described in OVP BHM and PPM API Functions Reference, which describes the parameters that control how the model interacts with the host computer.
Interrupts and FIFOs are supported.
Registers are aligned on 1 byte boundaries.

4.16.3 Limitations

Resolution of the baud rate is limited to the simulation time slice (aka quantum) size.
Values written to the MCR are ignored. Loopback mode is not supported.
The LSR is read-only. The model never sets the LSR 'Parity Error', 'Framing Error', 'Break Interrupt' or 'Error in RCVR FIFO' bits.
The MSR 'Data Set Ready' and 'Clear To Send' bits are set at reset and all other MSR bits are cleared. MSR bits will only be changed by writes to the MSR and values written to the Modem Status Register do not effect the operation of the model.

4.16.4 Reference

PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs datasheet (http://www.ti.com/lit/ds/symlink/pc16550d.pdf)

Table 23. Configuration options (attributes) set for instance 'uartTTY1'

Attribute	Value	Type	Expression
outfile	uartTTY1.log	string	
finishOnDisconnect	1	bool	

4.17 Peripheral [intel.ovpworld.org/peripheral/82077AA/1.0] instance: fd0

4.17.1 Licensing

Open Source Apache 2.0

4.17.2 Description

Dummy Floppy Disc Controller.

4.17.3 Limitations

Register stubs only.

4.17.4 Reference

http://www.buchty.net/casio/files/82077.pdf http://www.alldatasheet.com/datesheet-pdf/pdf/167793/INTEL/82077AA.html

There are no configuration options set for this peripheral instance.

4.18 Peripheral [mips.ovpworld.org/peripheral/16450C/1.0] instance: uartCBUS

4.18.1 Licensing
Open Source Apache 2.0

4.18.2 Description
Model of 16550/16450 UART.
Special version with register addresses for MIPS MALTA C-BUS.
Connects to a bus by a slave port and optionally to a processor by an interrupt signal.
The serial input/output ports are modelled by socket connection which must be attached to a process outside the simulation environment.
Note that on start:up, the UART model will block the simulator, pending a connection to the socket.

4.18.3 Limitations
No modelling of baud:rate.
No modem support (DTR etc).
No support for parity.
No means to simulate errors.

4.18.4 Reference
MIPS Malta Datasheet

Table 24. Configuration options (attributes) set for instance 'uartCBUS'

Attribute	Value	Type	Expression
outfile	uartCBUS.log	string	

4.19 Peripheral [mips.ovpworld.org/peripheral/MaltaFPGA/1.0] instance: maltaFpga

4.19.1 Licensing
Open Source Apache 2.0

4.19.2 Description
MIPS MALTA FPGA. Drives Development board functions.

4.19.3 Limitations
This model has sufficient functionality to allow a Linux Kernel to Boot on the MIPS:MALTA platform.

4.19.4 Reference
MIPS Malta User Manual.

Table 25. Configuration options (attributes) set for instance 'maltaFpga'

Attribute	Value	Type	Expression
stoponsoftreset	1	bool	

4.20 Peripheral [ovpworld.org/peripheral/Alpha2x16Display/1.0] instance: alphaDisplay

4.20.1 Description
This is a simple test peripheral creating a 2x16 alphanumeric display.

4.20.2 Licensing
Open Source Apache 2.0

4.20.3 Limitations
This is not representing a real device and provides simple operations as an example.

4.20.4 Reference
This is not based upon a real device

There are no configuration options set for this peripheral instance.

5.0 Overview of Imperas OVP Virtual Platforms

This document provides the details of the usage of an Imperas OVP Virtual Platform / Module. The first half of the document covers specifics of this particular virtual platform / module.

This second part of the document, includes information about Imperas OVP virtual platforms and modules, how they are built and used.

The Imperas virtual platforms are designed to provide a base for you to run high-speed software simulations of CPU-based SoCs and platforms on any suitable PC. They are typically based on the functionality of vendors fixed or evaluation platforms, enabling you to simulate software on these reference platforms. Typically virtual platforms are fixed and require the vendor to modify or extend them. Imperas virtual platforms are different in that they enable you to extend the functionality of the virtual platform, to closer reflect your own platform, by adding more component models, running different operating systems or adding additional applications.

Imperas virtual platforms are created using the Imperas iGen technology, allowing them to be used with Imperas OVP based simulators and also with Accellera/OSCI compliant SystemC simulators and commercial EDA System Design environments that use SystemC.

Virtual platforms include simulation models of the target devices, including the processor model(s) for the target device plus enough peripheral models to boot an operating system or run bare metal applications. The platform and the peripheral models used in most of the virtual platforms are open source, so that you can easily add new models to the platform as well as modify the existing models. Some models are only provided as binary, normally because the IP owner has restricted the release of the model source. In this case, please contact Imperas for more information.

There are typically several generic flavors of the virtual platforms for specific processor families, some targeting full operating systems, such as Linux, and some which focus on Real Time Operating Systems (RTOS) such as Mentor Nucleus or freeRTOS. OVP models of the processor cores are included in the virtual platforms, and for those processors which support mulitple cores SMP Linux is often supported for that virtual platform. For all of these virtual platforms, many of the peripheral components of the platform are modeled, often including the Ethernet and USB components. The semi-hosting capability of the Imperas virtual platform simulator products enables connection via the Ethernet and USB components from the virtual platform to the real world via the x86 host machine.

The Imperas OVP CPU models are written using the OVP Virtual Machine Interface (VMI) API that defines the behavior of the processor. The VMI API makes a clear line between model and simulator allowing very good optimization and world class high speed performance. The processor models are Instruction Accurate and do not model the detailed cycle timing of the processor and they implement functionality at the level of a Programmers View of the processor and peripherals and the software running on them does not know it is not running on hardware. Many models are provided as a binary shared object

and also as source. This allows the download and use of the model binary or the use of the source to explore and modify the model. The models are run through an extensive QA and regression testing process and most processor model families are validated using technology provided by the processor IP owners. All the models in this platform are developed with the Open Virtual Platforms APIs and are implemented in C. A platform can be modeled as different levels of hierarchy using separately describable and compilable modules.

More information on modeling and APIs can be found on the www.OVPworld.org site.

6.0 Getting Started with Imperas OVP Virtual Platforms

Virtual platforms are downloadable from the OVPworld website OVPworld.org/downloads. You need to browse and look for '<platform processor name> Examples'. You do need to be registered and logged in on the OVP site to download. OVPworld currently provides 32 bit host versions of packages containing virtual platforms.

When downloading, choose, Linux or Windows host. 32 bit packages can be installed and executed on 32 bit or 64 bit hosts. If you require a 64 bit host version please contact Imperas.

For example, for the ARM Versatile Express platform booting Linux on Cortex-A15MP Single, Dual, and Quad core procesors, you would want the download package: 'OVPsim_demo_Linux_ArmVersatileExpress_arm_Cortex-A15MP'.

Most virtual platform packages contain the platform and all the processor and peripheral models needed. You will need to download a simulator to run the platform. You can use OVPsim, downloadable from OVPworld.org/downloads, or you can use one of the Imperas simulators (imperas.com/products) available commercially from Imperas.

7.0 Simulating Software

7.1 Getting a license key to run

After you have downloaded you will need a runtime license key before the simulators will run. For OVPsim please visit OVPworld.org/likey and provide the required information and an evaluation/demo license key will be automatically sent to you. If you are using Imperas, then please contact Imperas for a license key.

7.2 Normal runs

To run a platform, read the section below on command line control of the platform and the section on setting command line arguments.

7.3 Loading Software

For most virtual platforms the platform is already configured to run the default software application/program and there is normally a script to run that sets some arguments. You can then copy/edit this script to select your own applications etc.

The example application programs are typically .elf format files and are provided pre-compiled. There are normally makefiles and associated scripts to recompile the example applications.

To find more information about compiling and loading software, the following document should be looked at: Imperas_Installation_and_Getting_Started.pdf.

7.4 Semihosting

In a virtual platform, semihosting is not normally used as there is normally hardware that implements the appropriate functionality - for example I/O will be handled by UARTs etc.

7.5 Using a terminal (UART)

If the platform includes one or more UARTs you will need to connect a terminal program to it so that you can see output and type into the simulated program. Review the list of peripherals below and see what configuration options it has been set with. In most cases there is an option to set to instruct the simulator to 'pop up' a terminal window connected to the simulated UART.

7.6 Interacting with the simulation (keyboard and mouse)

If the platform has a simulated UART you can normally set a command to get the simulator to pop up a terminal window allowing you to see output from the simulated UART and also allowing you to type characters into the UART that can be processed by the simulated software.

If your simulated platform has an LCD device then you can often configure it to recognize mouse movements and mouse clicks - allowing full interaction.

To see these interactions in action, have a look at some of the available videos available at OVPworld.org/demosandvideos.

7.7 More Information (Documentation) on Simulation

To find more information about running simulations and more of the options the simulators provide, the following documents should be looked at:
Imperas_Installation_and_Getting_Started.pdf
Simulation_Control_of_Platforms_and_Modules_User_Guide.pdf
Advanced_Simulation_Control_of_Platforms_and_Modules_User_Guide.pdf
OVP_Control_File_User_Guide.pdf

A full list of the currently available OVP documentation is available: OVPworld.org/documentation.

8.0 Debugging Software running on an Imperas OVP Virtual Platform

The Imperas and OVP simulators have several different interfaces to debuggers. These include several proprietary formats and also the standard GNU RSP format is supported allowing many compatible debuggers to be used. Below are some examples that Imperas directly support.

8.1 Debugging with GDB

A GNU debugger (GDB) can be connected to a processor in a platform using the RSP protocol. This allows the application program running on a processor to be debugged using a specific GDB for the processor selected. When using the Imperas Professional products many connections can be made allowing a GDB to be connected to all the processors in the platform.

The use of GDB is documented: OVPsim_Debugging_Applications_with_GDB_User_Guide.pdf.

*8.2 Debugging with Imperas M*DBG*

The Imperas multi-processor debugger can be connected to a platform and through this connection you can debug application programs running on all of the processors instanced within the platform. It is also capable, within this single unified environment, to debug peripheral model behavioral code in conjunction with the processor application programs.

For more information please see the Imperas M*DBG user guide.

The Imperas multi-processor debugger is also capable of controlling the Imperas Verification Analysis aand Profiling (VAP) tools in real time, making them invaluable to application program development, debugging and analysis.

For more information please see the Imperas VAP tools user guide.

8.3 Debugging with the Imperas eGui and GDB

Imperas eGui gives a GUI front end to the use of the GDB debugger. It allows use of all the features of GDB including source level application program debugging on processors.

*8.4 Debugging with the Imperas eGui and M*DBG*

Imperas eGui gives a GUI front end to the Imperas multi-processor debugger. It provides all the features of this debugger but does so with source level application program debugging on processors and source level debugging of the behavioral code on peripheral components in the platform. A context view shows all the processor and peripheral components within the platform and allows switching between them to examine the state of each at the event at which the simulation was stopped

Imperas eGui provides a menu from which the Imperas VAP tools can be controlled.

8.5 Debugging with Imperas eGui and Eclipse

Imperas provide a GUI based on Eclipse called eGui. This provides a GUI front end to use with a standard GDB or the Imperas MPD (Multi-Processor Debugger).

The use of eGui is documented: eGui_Eclipse_User_Guide.pdf.

A standard Eclipse CDT development environment can be connected to one or more processors in a

platform (multiple processors require an Imperas professional product). The simulation platform is started remotely or using the external tool feature in Eclipse, opens a debug port and awaits the connection with Eclipse. All features provided by the Eclipse CDT development environment are available to be used to debug software applications executing on the processors in the platform.

The use of Eclipse is documented: OVPsim_Debugging_Applications_with_Eclipse_User_Guide.pdf.

8.6 Debugging applications running under a simulated operating system

If the simulated platform is running an Operating System and the platform has a UART or Ethernet etc connection then it is often possible to connect an external debugger and debug the applications running under the simulated operating system.

An example would be a simulated platform running the Linux operating system, such as the MIPS Malta, or ARM Versatile Express. Within the simulated Linux you can start a gdbserver that connects from within the simulation through a UART out to the host PC via a port. Within the host PC you start a terminal program and connect to the port with a debugger such as GDB and can then debug the simulated user application.

9.0 Modifying the Platform / Module

9.1 Platforms / Modules use C/C++ and OVP APIs

The Imperas and OVP simulators execute a platform / module that is written in C/C++ and that makes function calls into the simulator's APIs. Thus the virtual platform / module is compiled from C/C++ into a binary shared object that the simulator loads and runs. OVP provides the definition and documentation that defines the C APIs for modeling the platforms, modules, the peripherals, and the processors. You can find more information about these APIs on the OVP website and in the OVP API documentation.

9.2 Platforms/Modules/Peripherals can be easily built with iGen from Imperas

Imperas provides a product 'iGen' that takes an input script file and creates the C/C++ files needed for platforms, modules, and peripherals - it creates the C/C++ file that is compiled into the platform, module or peripheral that is needed as an object file by the simulator. iGen creates the C/C++ files, you then need to add any necessary behaviors or further details etc. For platforms iGen creates either a C platform or a C++ SystemC TLM2 platform. For peripherals or modules iGen creates the C files and also provides a native C++ SystemC TLM2 interface to allow the peripheral/module to be instantiated in SystemC TLM2 platforms.

Information on iGen is available from: imperas.com/products.

9.3 Re-configuring the platform

There will nornmally be several configuration options that you can set when running the platform without the neeed to change any source. Refer to the section above on command line arguments. If these do not allow you to make the changes you need, then you may need to edit and recompile the source of the platform.

The source of the platform, modules, and the source of the peripherals will be installed as part of the packages you are using. The sources are located in the Imperas/OVP installation VLNV source tree. The VLNV term refers to: Vendor (eg arm.ovpworld.org), Library (eg platform), Name, (eg ArmVersatileExpress-CA15), and Version (eg 1.0). To modify the platform, locate the platform source files.

If you are an Imperas user and have access to iGen, we recommend you modify the source script files and regenerate and recompile the C that makes up the platform. Refer to the Imperas iGen model generator guide and the Imperas platform generator guide.

If you are using the C or SystemC TLM2 platforms with OVPsim, then you can edit the C/C++ files, recompile the source directly using the supplied makefiles, and the run the simulator directly with the resultant shared object.

9.4 Replacing peripherals components

If you need to replace peripherals, find the appropriate place in the source of the platform, make the change you need, and recompile etc. Look in the library for documentation on available peripherals and their configuration options.

9.5 Adding new peripherals components

If you need to add peripherals, find the appropriate place in the source, make the additions you need, and recompile etc. Look in the library for documentation on available peripherals and their configuration options.

If you need to create new peripheral components then use iGen to very quickly create the necessary C/C++ files that get you started. With iGen you can create peripherals with register/memory state in a few lines of iGen source. When adding behavior to the peripherals refer to the OVP API documentation.

10.0 Available Virtual Platforms

Table 26. Imperas / OVP Extendable Platform Kits (13 available)

Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
FreescaleKinetis60	freescale.ovpworld.org
FreescaleKinetis64	freescale.ovpworld.org
FreescaleVybridVFxx	freescale.ovpworld.org
MipsMalta	mips.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
XilinxML505	xilinx.ovpworld.org

Table 27. Imperas General Virtual Platforms (6 available)

Name	Vendor
arm-ti-eabi	arm.imperas.com
armm-ti-coff	arm.imperas.com
armm-ti-eabi	arm.imperas.com
HeteroAlteraCycloneV_HPS_CycloneIII_3c120	imperas.ovpworld.org
HeteroArmNucleusMIPSLinux	imperas.ovpworld.org
SiFiveFU540	imperas.ovpworld.org

Table 28. Imperas Modules (component of other platforms) (55 available)

Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
AE350	andes.ovpworld.org
ARMv8-A-FMv1	arm.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
ArmuKernel	imperas.ovpworld.org
ArmuKernelDual	imperas.ovpworld.org
BareMetalMIPS	imperas.ovpworld.org
Dual_ARMv8-A-FMv1_VLAN	imperas.ovpworld.org
Hetero_1xArm_3xMips32	imperas.ovpworld.org
Hetero_ARM_RISCV_NeuralNetwork	imperas.ovpworld.org

Hetero_ARMv8-A-FMv1_Cortex-M3	imperas.ovpworld.org
Hetero_ARMv8-A-FMv1_MIPS_microAptiv	imperas.ovpworld.org
Hetero_AlteraCycloneV_HPS_AlteraCycloneIII_3c120	imperas.ovpworld.org
Hetero_ArmIntegratorCP_XilinxMicroBlaze	imperas.ovpworld.org
Hetero_ArmVersatileExpress_MipsMalta	imperas.ovpworld.org
Hetero_ArmVersatileExpress_XilinxMicroBlaze	imperas.ovpworld.org
Quad_ArmVersatileExpress-CA15	imperas.ovpworld.org
RiscvRV32FreeRTOS	imperas.ovpworld.org
MipsMalta	mips.ovpworld.org
iMX6S	nxp.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org
virtio	riscv.ovpworld.org
FaultInjection	safepower.ovpworld.org
PublicDemonstrator	safepower.ovpworld.org
Zynq_PL_DualMicroblaze	safepower.ovpworld.org
Zynq_PL_NoC	safepower.ovpworld.org
Zynq_PL_NoC_node	safepower.ovpworld.org
Zynq_PL_NostrumNoC	safepower.ovpworld.org
Zynq_PL_NostrumNoC_node	safepower.ovpworld.org
Zynq_PL_RO	safepower.ovpworld.org
Zynq_PL_SingleMicroblaze	safepower.ovpworld.org
Zynq_PL_TTELNoC	safepower.ovpworld.org
Zynq_PL_TTELNoC_node	safepower.ovpworld.org
Zynq_PL_TTELNoC_processing_node_public_demonstrator	safepower.ovpworld.org
Zynq_PL_TTELNoC_public_demonstrator	safepower.ovpworld.org
Zynq_PL_TTELNoC_sensor_actor_node_public_demonstrator	safepower.ovpworld.org
FU540	sifive.ovpworld.org
S51CC	sifive.ovpworld.org
coreip-s51-arty	sifive.ovpworld.org
coreip-s51-rtl	sifive.ovpworld.org
dualFifo	vendor.com
XilinxML505	xilinx.ovpworld.org
Zynq	xilinx.ovpworld.org
Zynq_PL_Default	xilinx.ovpworld.org
Zynq_PS	xilinx.ovpworld.org
zc702	xilinx.ovpworld.org
zc706	xilinx.ovpworld.org

Table 29. Imperas / OVP Bare Metal Virtual Platforms (22 available)

Name	Vendor
BareMetalNios_IISingle	altera.ovpworld.org
BareMetalArcSingle	arc.ovpworld.org
BareMetalArm7Single	arm.ovpworld.org
BareMetalArmCortexADual	arm.ovpworld.org
BareMetalArmCortexASingle	arm.ovpworld.org
BareMetalArmCortexASingleAngelTrap	arm.ovpworld.org
BareMetalArmCortexMSingle	arm.ovpworld.org

ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
BareMetalArmx1Mips32x3	imperas.ovpworld.org
Or1kUclinux	imperas.ovpworld.org
BareMetalM14KSingle	mips.ovpworld.org
BareMetalMips32Dual	mips.ovpworld.org
BareMetalMips32Single	mips.ovpworld.org
BareMetalMips64Single	mips.ovpworld.org
BareMetalMipsDual	mips.ovpworld.org
BareMetalMipsSingle	mips.ovpworld.org
BareMetalOr1kSingle	ovpworld.org
BareMetalM16cSingle	posedgesoft.ovpworld.org
BareMetalPowerPc32Single	power.ovpworld.org
BareMetalV850Single	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org

#

