



Imperas Guide to using Virtual Platforms

Platform Specific Information for
arm.ovpworld.org / ARMv8-A-FMv1

Imperas Software Limited

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



Author	Imperas Software Limited
Version	20150901.0
Filename	Imperas_Platform_User_Guide_ARMv8-A-FMv1.pdf
Created	26 August 2015
Status	OVP Standard Release

Copyright Notice

Copyright 2015 Imperas Software Limited. All rights reserved. This software and documentation contain information that is the property of Imperas Software Limited. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Imperas Software Limited, or as expressly provided by the license agreement.

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 Virtual Platform: ARMv8-A-FMv1	5
1.1 Licensing	5
1.2 Description	5
1.3 Limitations	5
1.4 Reference	5
1.5 Location	5
1.6 Platform Simulation Attributes	5
2.0 Command Line Control of the Platform	6
2.1 Built-in Arguments	6
2.2 Platform Specific Command Line Arguments	6
3.0 Processor [arm.ovpworld.org/processor/arm/1.0] instance: cpu	6
3.1 Processor model type: 'arm' variant 'Cortex-A57MPx4' definition	6
3.2 Instance Parameters	8
3.3 Memory Map for processor 'cpu' bus: 'pBus'	8
3.4 Net Connections to processor: 'cpu'	9
4.0 Peripheral Instances	9
4.1 Peripheral [smc.ovpworld.org/peripheral/LAN91C111/1.0] instance: eth0	9
4.2 Peripheral [arm.ovpworld.org/peripheral/VexpressSysRegs/1.0] instance: sysRegs	10
4.3 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart0	10
4.4 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart1	10
4.5 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart2	11
4.6 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart3	11
4.7 Peripheral [ovpworld.org/peripheral/VirtioBlkMMIO/1.0] instance: vbd0	12
4.8 Peripheral [arm.ovpworld.org/peripheral/SmartLoaderArm64Linux/1.0] instance: smartLoader	12
5.0 Overview of Imperas OVP Virtual Platforms	14
6.0 Getting Started with Imperas OVP Virtual Platforms	15
7.0 Simulating Software	15
7.1 Getting a license key to run	15
7.2 Normal runs	15
7.3 Loading Software	15
7.4 Semihosting	16
7.5 Using a terminal (UART)	16
7.6 Interacting with the simulation (keyboard and mouse)	16
7.7 More Information (Documentation) on Simulation	16
8.0 Debugging Software running on an Imperas OVP Virtual Platform	16
8.1 Debugging with GDB	16
8.2 Debugging with Imperas M*DBG	17
8.3 Debugging with the Imperas iGui and GDB	17
8.4 Debugging with the Imperas iGui and M*DBG	17
8.5 Debugging with Eclipse	17

8.6 Debugging applications running under a simulated operating system	17
9.0 Modifying the Platform	18
9.1 Platforms use C/C++ and OVP APIs	18
9.2 Platforms/Peripherals can be easily built with iGen from Imperas	18
9.3 Re-configuring the platform	18
9.4 Replacing peripherals components	19
9.5 Adding new peripherals components	19
10.0 Available Virtual Platforms	20

1.0 Virtual Platform: ARMv8-A-FMv1

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

1.1 Licensing

Open Source Apache 2.0

1.2 Description

This platform implements the ARM v8-A Foundation Model v1 memory map described in ARM DUI 0677C.

The default processor is an ARM Cortex-A57MPx4.

The processor mips rate is modeled as 500MIPS by default.

The timerScaleFactor and processor MIPS rate default to values to model a 100MHz timer and CNTFREQ is automatically set accordingly.

This matches the clock frequency in the default Linux device tree. These should be adjusted if that is changed.

1.3 Limitations

This platform provides the peripherals required to boot and run Operating Systems such as Linux. Some of the peripherals are register-only, non-functional models. See the individual peripheral model documentation for details.

1.4 Reference

ARM DUI 0677C

1.5 Location

The ARMv8-A-FMv1 virtual platform is located in an Imperas/OVP installation at the VLNV: [arm.ovpworld.org / platform / ARMv8-A-FMv1 / 1.0](http://arm.ovpworld.org/platform/ARMv8-A-FMv1/1.0).

1.6 Platform Simulation Attributes

Table 1. Platform Simulation Attributes

Attribute	Value	Description
-----------	-------	-------------

stoponctrl	stoponctrl	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

No platform specific command line arguments have been specified.

3.0 Processor [arm.ovpworld.org/processor/arm/1.0] instance: cpu

3.1 Processor model type: 'arm' variant 'Cortex-A57MPx4' 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/arm.ovpworld.org/processor/arm/1.0/doc
- the OVP website: [OVP Model Specific Information arm Cortex-A57MPx4.pdf](#)

3.1.1 Description

ARM Processor Model

3.1.2 Licensing

Usage of binary model under license governing simulator usage.

Note that for models of ARM CPUs the license includes the following terms:

Licensee is granted a non-exclusive, worldwide, non-transferable, revocable licence to:

If no source is being provided to the Licensee: use and copy only (no modifications rights are granted) the model for the sole purpose of designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used to emulate an ARM based system to run application software in a production or live environment.

If source code is being provided to the Licensee: use, copy and modify the model for the sole purpose of

designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used to emulate an ARM based system to run application software in a production or live environment.

In the case of any Licensee who is either or both an academic or educational institution the purposes shall be limited to internal use.

Except to the extent that such activity is permitted by applicable law, Licensee shall not reverse engineer, decompile, or disassemble this model. If this model was provided to Licensee in Europe, Licensee shall not reverse engineer, decompile or disassemble the Model for the purposes of error correction.

The License agreement does not entitle Licensee to manufacture in silicon any product based on this model. The License agreement does not entitle Licensee to use this model for evaluating the validity of any ARM patent.

Source of model available under separate Imperas Software License Agreement.

ARMv8 architecture models additionally require a run time model license - contact Imperas for more information.

3.1.3 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. ISB, CP15ISB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. The model does not implement speculative fetch behavior. The branch cache is not modeled.

Caches and write buffers are not modeled in any way. All loads, fetches and stores complete immediately and in order, and are fully synchronous (as if the memory was of Strongly Ordered or Device-nGnRnE type). Data barrier instructions (e.g. DSB, CP15DSB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. Cache manipulation instructions are implemented as NOPs, with the exception of any undefined instruction behavior, which is modeled.

Real-world timing effects are not modeled: all instructions are assumed to complete in a single cycle.

Performance Monitors are implemented as a register interface only.

TLBs are architecturally-accurate but not device accurate. This means that all TLB maintenance and address translation operations are fully implemented but the cache is larger than in the real device.

Debug registers are implemented but non-functional (which is sufficient to allow operating systems such as Linux to boot). Debug state is not implemented.

This initial ARMv8 model also has the following specific limitations, which will be rectified shortly:

The model currently implements only a memory-mapped GICv2, not a register-accessible GICv3.

The optional CRC32 instructions are not supported.

The optional SIMD Cryptographic Extension instructions are not supported.

3.1.4 Verification

Models have been extensively tested by Imperas. ARM Cortex-A models have been successfully used by customers to simulate SMP Linux, Ubuntu Desktop, VxWorks and ThreadX on Xilinx Zynq virtual platforms.

3.1.5 Features

AArch64 is implemented at EL3, EL2, EL1 and EL0.

AArch32 is implemented at EL3, EL2, EL1 and EL0.

SIMD, VFP and LPA (large physical address extension) are implemented as standard in ARMv8.

Security extensions are implemented (also known as TrustZone). To make non-secure accesses visible externally, override ID_AA64MMFR0_EL1.PARange to specify the required physical bus size (32, 36, 40, 42, 44 or 48 bits) and connect the processor to a bus one bit wider (33, 37, 41, 43, 45 or 49 bits, respectively). The extra most-significant bit is the NS bit, indicating a non-secure access. If non-secure accesses are not required to be made visible externally, connect the processor to a bus of exactly the size implied by ID_AA64MMFR0_EL1.PARange.

VMSA EL1, EL2 and EL3 stage 1 address translation is implemented. VMSA stage 2 address translation is implemented.

Generic Timer is present. Use parameter override_timerScaleFactor to specify the counter rate as a fraction of the processor MIPS rate (e.g. 10 implies Generic Timer counters increment once every 10 processor instructions).

GIC block is implemented (GICv2, including security extensions). Accesses to GIC registers can be viewed externally by connecting to the 32-bit GICRegisters bus port. Secure register accesses are at offset 0x0 on this bus; for example, a secure access to GIC register GICD_CTLR can be observed by monitoring address 0x00001000. Non-secure accesses are at offset 0x80000000 on this bus; for example, a non-secure access to GIC register GICD_CTLR can be observed by monitoring address 0x80001000.

The internal GIC block can be disabled by raising signal GICCDISABLE, in which case the GIC needs to be modeled using a platform component instead. Input signals vfiq_CPU<N> and virq_CPU<N> can be used by an this component to raise virtual FIQ and IRQ interrupts on cores in the cluster if required.

3.2 Instance Parameters

Several parameters can be specified when a processor is instanced in a platform. For this processor instance 'cpu' it has been instanced with the following parameters:

Table 3. Processor Instance 'cpu' Parameters (Configurations)

Parameter	Value	Description
endian	little	Select processor endian (big or little)
simulateexceptions	simulateexceptions	Causes the processor simulate exceptions instead of halting
mips	500	The nominal MIPS for the processor

Table 4. Processor Instance 'cpu' Parameters (Attributes)

Parameter Name	Value	Type
variant	Cortex-A57MPx4	
compatibility	ISA	
UAL	1	
override_CBAR	0x2c000000	
override_GICD_TYPER_ITLines	4	
override_timerScaleFactor	5	

3.3 Memory Map for processor 'cpu' bus: 'pBus'

Processor instance 'cpu' is connected to bus 'pBus' using master port 'INSTRUCTION'.

Processor instance 'cpu' is connected to bus 'pBus' using master port 'DATA'.

Table 5. Memory Map ('cpu' / 'pBus' [width: 40])

Lo Address	Hi Address	Instance	Component
0x0	0x3FFFFFFF	RAM0	ram
0x4000000	0x403FFFFF	RAM1	ram
0x6000000	0x7FFFFFFF	RAM2	ram
0x1A000000	0x1A000FFF	eth0	LAN91C111
0x1C010000	0x1C010FFF	sysRegs	VexpressSysRegs
0x1C090000	0x1C090FFF	uart0	UartPL011
0x1C0A0000	0x1C0A0FFF	uart1	UartPL011
0x1C0B0000	0x1C0B0FFF	uart2	UartPL011
0x1C0C0000	0x1C0C0FFF	uart3	UartPL011
0x1C130000	0x1C1301FF	vbd0	VirtioBlkMMIO
0x80000000	0xFFFFFFFF	DRAM0	ram
0x880000000	0x9FFFFFFFFF	DRAM1	ram

3.4 Net Connections to processor: 'cpu'

Table 6. Processor Net Connections ('cpu')

Net Port	Net	Instance	Component
SPI37	ir5	uart0	UartPL011
SPI38	ir6	uart1	UartPL011
SPI39	ir7	uart2	UartPL011
SPI40	ir8	uart3	UartPL011
SPI47	ir15	eth0	LAN91C111
SPI74	ir42	vbd0	VirtioBlkMMIO

4.0 Peripheral Instances

4.1 Peripheral [smc.ovpworld.org/peripheral/LAN91C111/1.0] instance: eth0

4.1.1 Description

SMSC LAN91C111

4.1.2 Licensing

Open Source Apache 2.0

4.1.3 Limitations

Not all registers and device features are implemented. Only 16-bit bus interface currently supported.

4.1.4 Reference

SMSC LAN91C111 10/100 Non-PCI Ethernet Single Chip MAC + PHY Datasheet Revision 1.91 (06-01-09)

There are no configuration options set for this peripheral instance.

4.2 Peripheral [arm.ovpworld.org/peripheral/VexpressSysRegs/1.0] instance: sysRegs

4.2.1 Description

ARM Versatile Express System Registers

4.2.2 Limitations

Only select registers are modeled. See user.c for details.

4.2.3 Reference

A

4.2.4 Licensing

Open Source Apache 2.0

Table 7. Configuration options (attributes) set for instance 'sysRegs'

Attributes	Value
SYS_PROCID0	0x14000237

4.3 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart0

4.3.1 Description

ARM PL011 UART

4.3.2 Limitations

This is not a complete model of the PL011. There is no modeling of physical aspects of the UART, such as baud rates etc.

4.3.3 Reference

ARM PrimeCell UART (PL011) Technical Reference Manual (ARM DDI 0183)

4.3.4 Licensing

Open Source Apache 2.0

Table 8. Configuration options (attributes) set for instance 'uart0'

Attributes	Value
variant	ARM
outfile	uart0.log
finishOnDisconnect	1

4.4 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart1

4.4.1 Description

ARM PL011 UART

4.4.2 Limitations

This is not a complete model of the PL011. There is no modeling of physical aspects of the UART, such as baud rates etc.

4.4.3 Reference

ARM PrimeCell UART (PL011) Technical Reference Manual (ARM DDI 0183)

4.4.4 Licensing

Open Source Apache 2.0

Table 9. Configuration options (attributes) set for instance 'uart1'

Attributes	Value
variant	ARM
outfile	uart1.log

4.5 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart2

4.5.1 Description

ARM PL011 UART

4.5.2 Limitations

This is not a complete model of the PL011. There is no modeling of physical aspects of the UART, such as baud rates etc.

4.5.3 Reference

ARM PrimeCell UART (PL011) Technical Reference Manual (ARM DDI 0183)

4.5.4 Licensing

Open Source Apache 2.0

Table 10. Configuration options (attributes) set for instance 'uart2'

Attributes	Value
variant	ARM

4.6 Peripheral [arm.ovpworld.org/peripheral/UartPL011/1.0] instance: uart3

4.6.1 Description

ARM PL011 UART

4.6.2 Limitations

This is not a complete model of the PL011. There is no modeling of physical aspects of the UART, such as baud rates etc.

4.6.3 Reference

ARM PrimeCell UART (PL011) Technical Reference Manual (ARM DDI 0183)

4.6.4 Licensing

Open Source Apache 2.0

Table 11. Configuration options (attributes) set for instance 'uart3'

Attributes	Value
variant	ARM

4.7 Peripheral [ovpworld.org/peripheral/VirtioBlkMMIO/1.0] instance: *vbd0*

4.7.1 Description

VIRTIO version 1 mmio block device This model implements a VIRTIO MMIO block device as described in: <http://docs.oasis-open.org/virtio/virtio/v1.0/virtio-v1.0.pdf>. Use the VB_DRIVE parameter to specify the disk image file to use. Set the VB_DRIVE_DELTA parameter to 1 to prevent writes to disk during simulation from changing the image file.

4.7.2 Limitations

Only supports the Legacy (Device Version 1) interface. Only little endian guests are supported.

4.7.3 Licensing

Open Source Apache 2.0

4.7.4 Reference

<http://docs.oasis-open.org/virtio/virtio/v1.0/virtio-v1.0.pdf>

There are no configuration options set for this peripheral instance.

4.8 Peripheral [arm.ovpworld.org/peripheral/SmartLoaderArm64Linux/1.0] instance: *smartLoader*

4.8.1 Licensing

Open Source Apache 2.0

4.8.2 Description

Pseudo-peripheral to perform memory initialisation for an Arm64 Linux kernel boot: Loads Linux kernel image file, device tree blob and (optional) initial ram disk image into memory. Writes tiny boot code at

physical memory base to configure regs and then jump to the Kernel entry. Modifies the device tree to always use the spin-table enable-method.

4.8.3 Limitations

Only supports little endian

4.8.4 Reference

See ARM Linux boot requirements in Linux source tree at [documentation/arm64/booting.txt](#)

Table 12. Configuration options (attributes) set for instance 'smartLoader'

Attributes	Value
kernel	kernel
dtb	dtb
initrd	initrd
physicalbase	0x80000000
command	console=ttyAMA0 earlyprintk=pl011,0x1c090000 nokaslr
disable	disable

5.0 Overview of Imperas OVP Virtual Platforms

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

This second part of the document, includes information about Imperas OVP virtual platforms, 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 multiple 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.

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 processors, 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](#)

[OVPsim and CpuManager 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 and 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 iGui and GDB

Imperas iGui 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 iGui and M*DBG

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

8.5 Debugging with Eclipse

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

9.1 Platforms use C/C++ and OVP APIs

The Imperas and OVP simulators execute a platform that is written in C/C++ and that makes function calls into the simulator's APIs. Thus the virtual platform 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, 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/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 and peripherals - it creates the C/C++ file that is compiled into the platform 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 iGen creates the C files and also provides a native C++ SystemC TLM2 interface to allow the peripheral to be instantiated in SystemC TLM2 platforms.

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

9.3 Re-configuring the platform

There will normally be several configuration options that you can set when running the platform without the need 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 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 then 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 13. Imperas / OVP Extendable Platform Kits (17 available)

Platform Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
AlteraCycloneV_HPS_TLM2	altera.ovpworld.org
ARMv8-A-FMv1	arm.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmIntegratorCP_TLM2.0	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
ArmVersatileExpress_CA9_TLM2	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
FreescaleKinetis60	freescale.ovpworld.org
FreescaleVybridVF5	freescale.ovpworld.org
MipsMalta	mips.ovpworld.org
MipsMaltaLinux_TLM2.0	mips.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
XilinxML505	xilinx.ovpworld.org

Table 14. Imperas General Virtual Platforms (6 available)

Platform 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
QuadArmVersatileExpress	imperas.ovpworld.org

Table 15. Imperas / OVP Bare Metal Virtual Platforms (39 available)

Platform Name	Vendor
BareMetalNios_IISingle	altera.ovpworld.org
BareMetalNios_IISingle_TLM2.0	altera.ovpworld.org
BareMetalArcSingle	arc.ovpworld.org
BareMetalArcSingle_TLM2.0	arc.ovpworld.org
BareMetalArm7Single	arm.ovpworld.org
BareMetalArm7Single_TLM2.0	arm.ovpworld.org
BareMetalArmAArch64Single_TLM2.0	arm.ovpworld.org
BareMetalArmCortexADual	arm.ovpworld.org
BareMetalArmCortexASingle	arm.ovpworld.org
BareMetalArmCortexASingleAngelTrap	arm.ovpworld.org
BareMetalArmCortexASingle_TLM2.0	arm.ovpworld.org
BareMetalArmCortexMSingle	arm.ovpworld.org
BareMetalArmCortexMSingle_TLM2.0	arm.ovpworld.org

ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
BareMetalArcManycore24_TLM2.0	imperas.ovpworld.org
BareMetalArm7Dual_TLM2.0	imperas.ovpworld.org
BareMetalArmx1Mips32x3	imperas.ovpworld.org
BareMetalMips32Multicore2_TLM2.0	imperas.ovpworld.org
Or1kUclinux_TLM2.0	imperas.ovpworld.org
BareMetalM14KSingle	mips.ovpworld.org
BareMetalM14KSingle_TLM2.0	mips.ovpworld.org
BareMetalMips32Dual	mips.ovpworld.org
BareMetalMips32Single	mips.ovpworld.org
BareMetalMips32Single_TLM2.0	mips.ovpworld.org
BareMetalMips64Single	mips.ovpworld.org
BareMetalMips64Single_TLM2.0	mips.ovpworld.org
BareMetalMipsDual	mips.ovpworld.org
BareMetalMipsSingle	mips.ovpworld.org
BareMetalMipsSingle_TLM2.0	mips.ovpworld.org
BareMetalOr1kSingle	ovpworld.org
BareMetalOr1kSingle_TLM2.0	ovpworld.org
BareMetalM16cSingle	posedgesoft.ovpworld.org
BareMetalPowerPc32Single	power.ovpworld.org
BareMetalPowerPc32Single_TLM2.0	power.ovpworld.org
BareMetalV850Single	renesas.ovpworld.org
BareMetalV850Single_TLM2.0	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org
BareMetalMicroBlazeSingle_TLM2.0	xilinx.ovpworld.org

#