



## Imperas Guide to using Virtual Platforms

Platform / Module Specific Information for  
[riscv.ovpworld.org](http://riscv.ovpworld.org) / virtio

### Imperas Software Limited

Imperas Buildings, North Weston  
Thame, Oxfordshire, OX9 2HA, U.K.  
[docs@imperas.com](mailto:docs@imperas.com).



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

## Copyright Notice

Copyright 2021 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](http://OVPworld.org).

## Table Of Contents

<b>1.0 Platform / Module: virtio</b>	5
1.1 Virtual Platform / Module Type	5
1.2 Licensing	5
1.3 Description	5
1.4 Reference	5
1.5 Limitations	5
1.6 Location	5
1.7 Module Simulation Attributes	5
<b>2.0 Processor [riscv.ovpworld.org/processor/riscv/1.0] instance: hart0</b>	5
2.1 Processor model type: 'riscv' variant 'RVB64I' definition	5
2.2 Instance Parameters	14
2.3 Memory Map for processor 'hart0' bus: 'bus0'	14
2.4 Net Connections to processor: 'hart0'	15
<b>3.0 Peripheral Instances</b>	15
3.1 Peripheral [ovpworld.org/peripheral/VirtioBlkMMIO/1.0] instance: VBD0	15
3.2 Peripheral [ovpworld.org/peripheral/VirtioNetMMIO/1.0] instance: VND0	15
3.3 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio_mmio3	16
3.4 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio_mmio4	16
3.5 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio_mmio5	17
3.6 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio_mmio6	17
3.7 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio_mmio7	18
3.8 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio_mmio8	18
3.9 Peripheral [riscv.ovpworld.org/peripheral/CLINT/1.0] instance: clint	19
3.10 Peripheral [riscv.ovpworld.org/peripheral/PLIC/1.0] instance: plic	19
3.11 Peripheral [national.ovpworld.org/peripheral/16550/1.0] instance: uartTTY0	20
3.12 Peripheral [riscv.ovpworld.org/peripheral/SmartLoaderRV64Linux/1.0] instance: smartLoader	20
<b>4.0 Overview of Imperas OVP Virtual Platforms</b>	22
<b>5.0 Getting Started with Imperas OVP Virtual Platforms</b>	23
<b>6.0 Simulating Software</b>	23
6.1 Getting a license key to run	23
6.2 Normal runs	23
6.3 Loading Software	23
6.4 Semihosting	24
6.5 Using a terminal (UART)	24
6.6 Interacting with the simulation (keyboard and mouse)	24
6.7 More Information (Documentation) on Simulation	24
<b>7.0 Debugging Software running on an Imperas OVP Virtual Platform</b>	24
7.1 Debugging with GDB	24
7.2 Debugging with Imperas M*DBG	25
7.3 Debugging with the Imperas eGui and GDB	25

7.4 Debugging with the Imperas eGui and M*DBG	25
7.5 Debugging with Imperas eGui and Eclipse	25
7.6 Debugging applications running under a simulated operating system	26
<b>8.0 Modifying the Platform / Module</b>	<b>26</b>
8.1 Platforms / Modules use C/C++ and OVP APIs	26
8.2 Platforms/Modules/Peripherals can be easily built with iGen from Imperas	26
8.3 Re-configuring the platform	26
8.4 Replacing peripherals components	27
8.5 Adding new peripherals components	27
<b>9.0 Available Virtual Platforms</b>	<b>28</b>

## 1.0 Platform / Module: virtio

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 hardware component is described as being a module. A module is a component that is used in other modules, platforms, or test harnesses. It is normally used to encapsulate a layer in a hierarchical system.

### 1.2 Licensing

Open Source Apache 2.0

### 1.3 Description

Virtio System Platform to boot BusyBear-Linux Kernel

### 1.4 Reference

Virtio System Platform

### 1.5 Limitations

Sufficient functionality to boot BusyBear-Linux Kernel using the Virtio platform

### 1.6 Location

The virtio virtual platform / module is located in an Imperas/OVP installation at the VLNV:  
[riscv.ovpworld.org / module / virtio / 1.0](http://riscv.ovpworld.org/module/virtio/1.0).

### 1.7 Module Simulation Attributes

Table 1. Module Simulation Attributes

Attribute	Value	Description
stoponctrlc	stoponctrlc	Stop on control-C

## 2.0 Processor [[riscv.ovpworld.org/processor/riscv/1.0](http://riscv.ovpworld.org/processor/riscv/1.0)] instance: hart0

### 2.1 Processor model type: 'riscv' variant 'RVB64I' 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/riscv.ovpworld.org/processor/riscv/1.0/doc](http://ImperasLib/source/riscv.ovpworld.org/processor/riscv/1.0/doc)
- the OVP website: [OVP\\_Model\\_Specific\\_Information\\_riscv\\_RVB64I.pdf](#)

#### 2.1.1 Description

RISC-V RVB64I 64-bit processor model

### 2.1.2 Licensing

This Model is released under the Open Source Apache 2.0

### 2.1.3 Extensions Enabled by Default

The model has the following architectural extensions enabled, and the corresponding bits in the misa CSR Extensions field will be set upon reset:

misa bit 8: RV32I/RV64I/RV128I base integer instruction set

To specify features that can be dynamically enabled or disabled by writes to the misa register in addition to those listed above, use parameter "add\_Extensions\_mask". This is a string parameter containing the feature letters to add; for example, value "DV" indicates that double-precision floating point and the Vector Extension can be enabled or disabled by writes to the misa register, if supported on this variant. Parameter "sub\_Extensions\_mask" can be used to disable dynamic update of features in the same way.

Legacy parameter "misa\_Extensions\_mask" can also be used. This Uns32-valued parameter specifies all writable bits in the misa Extensions field, replacing any permitted bits defined in the base variant.

Note that any features that are indicated as present in the misa mask but absent in the misa will be ignored. See the next section.

### 2.1.4 Enabling Other Extensions

The following extensions are supported by the model, but not enabled by default in this variant:

misa bit 0: extension A (atomic instructions)

misa bit 1: extension B (bit manipulation extension)

misa bit 2: extension C (compressed instructions)

misa bit 3: extension D (double-precision floating point)

misa bit 4: RV32E base integer instruction set (embedded)

misa bit 5: extension F (single-precision floating point)

misa bit 7: extension H (hypervisor)

misa bit 10: extension K (cryptographic)

misa bit 12: extension M (integer multiply/divide instructions)

misa bit 13: extension N (user-level interrupts)

misa bit 15: extension P (DSP instructions)

misa bit 18: extension S (Supervisor mode)

misa bit 20: extension U (User mode)

misa bit 21: extension V (vector extension)

misa bit 23: extension X (non-standard extensions present)

To add features from this list to the visible set in the misa register, use parameter "add\_Extensions". This is a string containing identification letters of features to enable; for example, value "DV" indicates that double-precision floating point and the Vector Extension should be enabled, if they are currently absent and are available on this variant.

Legacy parameter "misa\_Extensions" can also be used. This Uns32-valued parameter specifies the reset value for the misa CSR Extensions field, replacing any permitted bits defined in the base variant.

To add features from this list to the implicitly-enabled set (not visible in the misa register), use parameter "add\_implicit\_Extensions". This is a string parameter in the same format as the "add\_Extensions" parameter described above.

### 2.1.5 mtvec CSR

On this variant, the Machine trap-vector base-address register (mtvec) is writable. It can instead be configured as read-only using parameter "mtvec\_is\_ro".

Values written to "mtvec" are masked using the value 0xffffffffffffd. A different mask of writable bits may be specified using parameter "mtvec\_mask" if required. In addition, when Vectored interrupt mode is enabled, parameter "tvec\_align" may be used to specify additional hardware-enforced base address alignment. In this variant, "tvec\_align" defaults to 0, implying no alignment constraint.

If parameter "mtvec\_sext" is True, values written to "mtvec" are sign-extended from the most-significant writable bit. In this variant, "mtvec\_sext" is False, indicating that "mtvec" is not sign-extended.

The initial value of "mtvec" is 0x0. A different value may be specified using parameter "mtvec" if required.

### 2.1.6 Reset

On reset, the model will restart at address 0x0. A different reset address may be specified using parameter "reset\_address" or applied using optional input port "reset\_addr" if required.

### 2.1.7 NMI

On an NMI, the model will restart at address 0x0; a different NMI address may be specified using parameter "nmi\_address" or applied using optional input port "nmi\_addr" if required. The cause reported on an NMI is 0x0 by default; a different cause may be specified using parameter "ecode\_nmi" or applied using optional input port "nmi\_cause" if required.

If parameter "rnmi\_version" is not "none", resumable NMIs are supported, managed by additional CSRs "mnscratch", "mnepc", "mncause" and "mnstatus", following the indicated version of the Resumable NMI extension proposal. In this variant, "rnmi\_version" is "none".

### 2.1.8 WFI

WFI will halt the processor until an interrupt occurs. It can instead be configured as a NOP using parameter "wfi\_is\_nop". WFI timeout wait is implemented with a time limit of 0 (i.e. WFI causes an Illegal Instruction trap in Supervisor mode when mstatus.TW=1).

### 2.1.9 cycle CSR

The "cycle" CSR is implemented in this variant. Set parameter "cycle\_undefined" to True to instead specify that "cycle" is unimplemented and reads of it should cause Illegal Instruction traps.

### 2.1.10 time CSR

The "time" CSR is implemented in this variant. Set parameter "time\_undefined" to True to instead specify that "time" is unimplemented and reads of it should cause Illegal Instruction traps. Usually, the value of the "time" CSR should be provided by the platform - see notes below about the artifact "CSR" bus for information about how this is done.

### 2.1.11 instret CSR

The "instret" CSR is implemented in this variant. Set parameter "instret\_undefined" to True to instead specify that "instret" is unimplemented and reads of it should cause Illegal Instruction traps.

#### 2.1.12 hpmcounter CSRs

"hpmcounter" CSRs are implemented in this variant. Set parameter "hpmcounter\_undefined" to True to instead specify that "hpmcounter" CSRs are unimplemented and reads of them should cause Illegal Instruction traps.

#### 2.1.13 Unaligned Accesses

Unaligned memory accesses are not supported by this variant. Set parameter "unaligned" to "T" to enable such accesses.

#### 2.1.14 PMP

A PMP unit is not implemented by this variant. Set parameter "PMP\_registers" to indicate that the unit should be implemented with that number of PMP entries.

#### 2.1.15 Privileged Architecture

This variant implements the Privileged Architecture with version specified in the References section of this document. Note that parameter "priv\_version" can be used to select the required architecture version; see the following sections for detailed information about differences between each supported version.

#### 2.1.16 Legacy Version 1.10

1.10 version of May 7 2017.

#### 2.1.17 Version 20190608

Stable 1.11 version of June 8 2019, with these changes compared to version 1.10:

- mcountinhibit CSR defined;
- pages are never executable in Supervisor mode if page table entry U bit is 1;
- mstatus.TW is writable if any lower-level privilege mode is implemented (previously, it was just if Supervisor mode was implemented);

#### 2.1.18 Version master

Unstable master version corresponding to evolving 1.12 specification, with these changes compared to version 20190608:

- mstatush, mseccfg, mseccfgh, menvcfg, menvcfgh, senvcfg, henvcfg, henvcfgh and mconfigptr CSRs defined;
- xret instructions clear mstatus.MPRV when leaving Machine mode if new mode is less privileged than M-mode;
- maximum number of PMP registers increased to 64;
- data endian is now configurable.

#### 2.1.19 Unprivileged Architecture

This variant implements the Unprivileged Architecture with version specified in the References section of this document. Note that parameter "user\_version" can be used to select the required architecture version; see the following sections for detailed information about differences between each supported version.



### 2.1.20 Legacy Version 2.2

2.2 version of May 7 2017.

### 2.1.21 Version 20191213

Stable 20191213-Base-Ratified version of December 13 2019, with these changes compared to version 2.2:

- floating point fmin/fmax instruction behavior modified to comply with IEEE 754-201x.
- numerous other optional behaviors can be separately enabled using Z-prefixed parameters.

### 2.1.22 Other Extensions

Other extensions that can be configured are described in this section.

### 2.1.23 Zicsr

Parameter "Zicsr" is 1 on this variant, meaning that standard CSRs and CSR access instructions are implemented. if "Zicsr" is set to 0 then standard CSRs and CSR access instructions are not implemented and an alternative scheme must be provided as a processor extension.

### 2.1.24 Zifencei

Parameter "Zifencei" is 1 on this variant, meaning that the fence.i instruction is implemented (but treated as a NOP by the model). if "Zifencei" is set to 0 then the fence.i instruction is not implemented.

### 2.1.25 Zicbom

Parameter "Zicbom" is 0 on this variant, meaning that code block management instructions are undefined. if "Zicbom" is set to 1 then code block management instructions cbo.clean, cbo.flush and cbo.inval are defined.

If Zicbom is present, the cache block size is given by parameter "cmomp\_bytes". The instructions may cause traps if used illegally but otherwise are NOPs in this model.

### 2.1.26 Zicbop

Parameter "Zicbop" is 0 on this variant, meaning that prefetch instructions are undefined. if "Zicbop" is set to 1 then prefetch instructions prefetch.i, prefetch.r and prefetch.w are defined (but behave as NOPs in this model).

### 2.1.27 Zicboz

Parameter "Zicboz" is 0 on this variant, meaning that the cbo.zero instruction is undefined. if "Zicboz" is set to 1 then the cbo.zero instruction is defined.

If Zicboz is present, the cache block size is given by parameter "cmoz\_bytes".

### 2.1.28 CLIC

The model can be configured to implement a Core Local Interrupt Controller (CLIC) using parameter "CLICLEVELS"; when non-zero, the CLIC is present with the specified number of interrupt levels (2-256), as described in the RISC-V Core-Local Interrupt Controller specification, and further parameters are made available to configure other aspects of the CLIC. "CLICLEVELS" is zero in this variant, indicating that a CLIC is not implemented.

### 2.1.29 Interrupts

The "reset" port is an active-high reset input. The processor is halted when "reset" goes high and resumes execution from the reset address specified using the "reset\_address" parameter or "reset\_addr" port when the signal goes low. The "mcause" register is cleared to zero.

The "nmi" port is an active-high NMI input. The processor resumes execution from the address specified using the "nmi\_address" parameter or "nmi\_addr" port when the NMI signal goes high. The "mcause" register is cleared to zero.

All other interrupt ports are active high. For each implemented privileged execution level, there are by default input ports for software interrupt, timer interrupt and external interrupt; for example, for Machine mode, these are called "MSWInterrupt", "MTimerInterrupt" and "MExternalInterrupt", respectively. When the N extension is implemented, ports are also present for User mode. Parameter "unimp\_int\_mask" allows the default behavior to be changed to exclude certain interrupt ports. The parameter value is a mask in the same format as the "mip" CSR; any interrupt corresponding to a non-zero bit in this mask will be removed from the processor and read as zero in "mip", "mie" and "mideleg" CSRs (and Supervisor and User mode equivalents if implemented).

Parameter "external\_int\_id" can be used to enable extra interrupt ID input ports on each hart. If the parameter is True then when an external interrupt is applied the value on the ID port is sampled and used to fill the Exception Code field in the "mcause" CSR (or the equivalent CSR for other execution levels). For Machine mode, the extra interrupt ID port is called "MExternalInterruptID".

The "deferint" port is an active-high artifact input that, when written to 1, prevents any pending-and-enabled interrupt being taken (normally, such an interrupt would be taken on the next instruction after it becomes pending-and-enabled). The purpose of this signal is to enable alignment with hardware models in step-and-compare usage.

### 2.1.30 Debug Mode

The model can be configured to implement Debug mode using parameter "debug\_mode". This implements features described in Chapter 4 of the RISC-V External Debug Support specification with version specified by parameter "debug\_version" (see References). Some aspects of this mode are not defined in the specification because they are implementation-specific; the model provides infrastructure to allow implementation of a Debug Module using a custom harness. Features added are described below.

Parameter "debug\_mode" can be used to specify three different behaviors, as follows:

1. If set to value "vector", then operations that would cause entry to Debug mode result in the processor jumping to the address specified by the "debug\_address" parameter. It will execute at this address, in Debug mode, until a "dret" instruction causes return to non-Debug mode. Any exception generated during this execution will cause a jump to the address specified by the "dexc\_address" parameter.
2. If set to value "interrupt", then operations that would cause entry to Debug mode result in the processor simulation call (e.g. opProcessorSimulate) returning, with a stop reason of OP\_SR\_INTERRUPT. In this usage scenario, the Debug Module is implemented in the simulation harness.
3. If set to value "halt", then operations that would cause entry to Debug mode result in the processor halting. Depending on the simulation environment, this might cause a return from the simulation call with a stop reason of OP\_SR\_HALT, or debug mode might be implemented by another platform component which then restarts the debugged processor again.

### 2.1.31 Debug State Entry

The specification does not define how Debug mode is implemented. In this model, Debug mode is enabled by a Boolean pseudo-register, "DM". When "DM" is True, the processor is in Debug mode. When "DM" is False, mode is defined by "mstatus" in the usual way.

Entry to Debug mode can be performed in any of these ways:

1. By writing True to register "DM" (e.g. using opProcessorRegWrite) followed by simulation of at least one cycle (e.g. using opProcessorSimulate), dcsr cause will be reported as trigger;
2. By writing a 1 then 0 to net "haltreq" (using opNetWrite) followed by simulation of at least one cycle (e.g. using opProcessorSimulate);
3. By writing a 1 to net "resethaltreq" (using opNetWrite) while the "reset" signal undergoes a negedge transition, followed by simulation of at least one cycle (e.g. using opProcessorSimulate);
4. By executing an "ebreak" instruction when Debug mode entry for the current processor mode is enabled by dcsr.ebreakm, dcsr.ebreaks or dcsr.ebreaku.

In all cases, the processor will save required state in "dpc" and "dcsr" and then perform actions described above, depending in the value of the "debug\_mode" parameter.

### 2.1.32 Debug State Exit

Exit from Debug mode can be performed in any of these ways:

1. By writing False to register "DM" (e.g. using opProcessorRegWrite) followed by simulation of at least one cycle (e.g. using opProcessorSimulate);
2. By executing an "dret" instruction when Debug mode.

In both cases, the processor will perform the steps described in section 4.6 (Resume) of the Debug specification.

### 2.1.33 Debug Registers

When Debug mode is enabled, registers "dcsr", "dpc", "dscratch0" and "dscratch1" are implemented as described in the specification. These may be manipulated externally by a Debug Module using opProcessorRegRead or opProcessorRegWrite; for example, the Debug Module could write "dcsr" to enable "ebreak" instruction behavior as described above, or read and write "dpc" to emulate stepping over an "ebreak" instruction prior to resumption from Debug mode.

### 2.1.34 Debug Mode Execution

The specification allows execution of code fragments in Debug mode. A Debug Module implementation can cause execution in Debug mode by the following steps:

1. Write the address of a Program Buffer to the program counter using opProcessorPCSet;
2. If "debug\_mode" is set to "halt", write 0 to pseudo-register "DMStall" (to leave halted state);
3. If entry to Debug mode was handled by exiting the simulation callback, call opProcessorSimulate or opRootModuleSimulate to resume simulation.

Debug mode will be re-entered in these cases:

1. By execution of an "ebreak" instruction; or
2. By execution of an instruction that causes an exception.

In both cases, the processor will either jump to the debug exception address, or return control immediately to the harness, with stopReason of OP\_SR\_INTERRUPT, or perform a halt, depending on the value of the

"debug\_mode" parameter.

#### **2.1.35 Debug Single Step**

When in Debug mode, the processor or harness can cause a single instruction to be executed on return from that mode by setting `dcscr.step`. After one non-Debug-mode instruction has been executed, control will be returned to the harness. The processor will remain in single-step mode until `dcscr.step` is cleared.

#### **2.1.36 Debug Ports**

Port "DM" is an output signal that indicates whether the processor is in Debug mode

Port "haltreq" is a rising-edge-triggered signal that triggers entry to Debug mode (see above).

Port "resethaltreq" is a level-sensitive signal that triggers entry to Debug mode after reset (see above).

#### **2.1.37 Trigger Module**

This model is configured with a trigger module, implementing a subset of the behavior described in Chapter 5 of the RISC-V External Debug Support specification with version specified by parameter "debug\_version" (see References).

#### **2.1.38 Trigger Module Restrictions**

The model currently supports `tdata1` of type 0, type 2 (`mcontrol`), type 3 (`icount`), type 4 (`itrigger`), type 5 (`etrigger`) and type 6 (`mcontrol6`). `icount` triggers are implemented for a single instruction only, with count hard-wired to 1 and automatic zeroing of mode bits when the trigger fires.

#### **2.1.39 Trigger Module Parameters**

Parameter "trigger\_num" is used to specify the number of implemented triggers. In this variant, "trigger\_num" is 4.

Parameter "tinfo" is used to specify the value of the read-only "tinfo" register, which indicates the trigger types supported. In this variant, "tinfo" is 0x7d.

Parameter "tinfo\_undefined" is used to specify whether the "tinfo" register is undefined, in which case reads of it trap to Machine mode. In this variant, "tinfo\_undefined" is 0.

Parameter "tcontrol\_undefined" is used to specify whether the "tcontrol" register is undefined, in which case accesses to it trap to Machine mode. In this variant, "tcontrol\_undefined" is 0.

Parameter "mcontext\_undefined" is used to specify whether the "mcontext" register is undefined, in which case accesses to it trap to Machine mode. In this variant, "mcontext\_undefined" is 0.

Parameter "scontext\_undefined" is used to specify whether the "scontext" register is undefined, in which case accesses to it trap to Machine mode. In this variant, "scontext\_undefined" is 0.

Parameter "mscontext\_undefined" is used to specify whether the "mscontext" register is undefined, in which case accesses to it trap to Machine mode. In this variant, "mscontext\_undefined" is 0.

Parameter "amo\_trigger" is used to specify whether load/store triggers are activated for AMO instructions. In this variant, "amo\_trigger" is 0.

Parameter "no\_hit" is used to specify whether the "hit" bit in `tdata1` is unimplemented. In this variant, "no\_hit" is 0.

Parameter "mcontext\_bits" is used to specify the number of writable bits in the "mcontext" register. In this variant, "mcontext\_bits" is 13.

Parameter "mvalue\_bits" is used to specify the number of writable bits in the "mvalue" field in "extra32"/"extra64" registers; if zero, the "mselect" field is tied to zero. In this variant, "mvalue\_bits" is 13.

Parameter "mcontrol\_maskmax" is used to specify the value of field "maskmax" in the "mcontrol" register. In this variant, "mcontrol\_maskmax" is 63.

#### **2.1.40 Debug Mask**

It is possible to enable model debug messages in various categories. This can be done statically using the "override\_debugMask" parameter, or dynamically using the "debugflags" command. Enabled messages are specified using a bitmask value, as follows:

Value 0x002: enable debugging of PMP and virtual memory state;

Value 0x004: enable debugging of interrupt state.

All other bits in the debug bitmask are reserved and must not be set to non-zero values.

#### **2.1.41 Integration Support**

This model implements a number of non-architectural pseudo-registers and other features to facilitate integration.

#### **2.1.42 CSR Register External Implementation**

If parameter "enable\_CSR\_bus" is True, an artifact 16-bit bus "CSR" is enabled. Slave callbacks installed on this bus can be used to implement modified CSR behavior (use opBusSlaveNew or icmMapExternalMemory, depending on the client API). A CSR with index 0xABC is mapped on the bus at address 0xABC0; as a concrete example, implementing CSR "time" (number 0xC01) externally requires installation of callbacks at address 0xC010 on the CSR bus.

#### **2.1.43 Limitations**

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. fence.i) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is 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. Data barrier instructions (e.g. fence) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is modeled.

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

Hardware Performance Monitor registers are not implemented and hardwired to zero.

#### **2.1.44 Verification**

All instructions have been extensively tested by Imperas, using tests generated specifically for this model and also reference tests from <https://github.com/riscv/riscv-tests>.

Also reference tests have been used from various sources including:

<https://github.com/riscv/riscv-tests>

<https://github.com/ucb-bar/riscv-torture>

The Imperas OVPsim RISC-V models are used in the RISC-V Foundation Compliance Framework as a functional Golden Reference:

<https://github.com/riscv/riscv-compliance>

where the simulated model is used to provide the reference signatures for compliance testing. The Imperas OVPsim RISC-V models are used as reference in both open source and commercial instruction stream test generators for hardware design verification, for example:

<http://valtrix.in/sting> from Valtrix

<https://github.com/google/riscv-dv> from Google

The Imperas OVPsim RISC-V models are also used by commercial and open source RISC-V Core RTL developers as a reference to ensure correct functionality of their IP.

### 2.1.45 References

The Model details are based upon the following specifications:

RISC-V Instruction Set Manual, Volume I: User-Level ISA (User Architecture Version 20191213)

RISC-V Instruction Set Manual, Volume II: Privileged Architecture (Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11)

### 2.2 Instance Parameters

Several parameters can be specified when a processor is instanced in a platform. For this processor instance 'hart0' it has been instanced with the following parameters:

Table 2. Processor Instance 'hart0' Parameters (Configurations)

Parameter	Value	Description
simulateexceptions	simulateexceptions	Causes the processor simulate exceptions instead of halting
mips	50	The nominal MIPS for the processor

Table 3. Processor Instance 'hart0' Parameters (Attributes)

Parameter Name	Value	Type
variant	RVB64I	enum
add_Extensions	MAFDCSU	string
reset_address	0x1000	Uns64
time_undefined	True	bool

### 2.3 Memory Map for processor 'hart0' bus: 'bus0'

Processor instance 'hart0' is connected to bus 'bus0' using master port 'INSTRUCTION'.

Processor instance 'hart0' is connected to bus 'bus0' using master port 'DATA'.

Table 4. Memory Map ( 'hart0' / 'bus0' [width: 48] )

Lo Address	Hi Address	Instance	Component
remappable	remappable	virtio_mmio3	trap
remappable	remappable	virtio_mmio4	trap
remappable	remappable	virtio_mmio5	trap
remappable	remappable	virtio_mmio6	trap
remappable	remappable	virtio_mmio7	trap
remappable	remappable	virtio_mmio8	trap
0x1000	0x3FFF	bootram	ram

0x2000000	0x200BFFF	clint	CLINT
0xC000000	0xFFFFFFFF	plic	PLIC
0x10000000	0x10000007	uartTTY0	16550
0x10001000	0x100011FF	VBD0	VirtioBlkMMIO
0x10002000	0x100021FF	VND0	VirtioNetMMIO
0x80000000	0x87FFFFFFF	main	ram

## 2.4 Net Connections to processor: 'hart0'

Table 5. Processor Net Connections ( 'hart0' )

Net Port	Net	Instance	Component
MExternalInterrupt	irqT0	plic	PLIC
SExternalInterrupt	irqT1	plic	PLIC
MTimerInterrupt	MTimerInterrupt0	clint	CLINT
MSWInterrupt	MSWInterrupt0	clint	CLINT

## 3.0 Peripheral Instances

### 3.1 Peripheral [ovpworld.org/peripheral/VirtioBlkMMIO/1.0] instance: VBD0

#### 3.1.1 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.

#### 3.1.2 Limitations

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

#### 3.1.3 Licensing

Open Source Apache 2.0

#### 3.1.4 Reference

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

There are no configuration options set for this peripheral instance.

### 3.2 Peripheral [ovpworld.org/peripheral/VirtioNetMMIO/1.0] instance: VND0

#### 3.2.1 Description

VIRTIO version 1 mmio block device This model implements a VIRTIO MMIO net device as described in: <http://docs.oasis-open.org/virtio/virtio/v1.0/virtio-v1.0.pdf>.

### 3.2.2 Limitations

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

### 3.2.3 Licensing

Open Source Apache 2.0

### 3.2.4 Reference

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

There are no configuration options set for this peripheral instance.

## 3.3 Peripheral [[ovpworld.org/peripheral/trap/1.0](http://ovpworld.org/peripheral/trap/1.0)] instance: *virtio\_mmio3*

### 3.3.1 Description

Open a port and allocate a region that is defined by parameters.

The region can be configured to act as standard memory or can report read/write accesses.

### 3.3.2 Licensing

Open Source Apache 2.0

### 3.3.3 Limitations

This peripheral cannot be used in a hardware description used to generate a TLM platform.

### 3.3.4 Reference

This is not based upon the operation of a real device but is intended to be used for bring up and development of new virtual platforms.

Table 6. Configuration options (attributes) set for instance 'virtio\_mmio3'

Attribute	Value	Type	Expression
portAddress	0x10003000	uns32	
portSize	4096	uns32	
cbEnable	True	boolean	

## 3.4 Peripheral [[ovpworld.org/peripheral/trap/1.0](http://ovpworld.org/peripheral/trap/1.0)] instance: *virtio\_mmio4*

### 3.4.1 Description

Open a port and allocate a region that is defined by parameters.

The region can be configured to act as standard memory or can report read/write accesses.

### 3.4.2 Licensing

Open Source Apache 2.0

### 3.4.3 Limitations



This peripheral cannot be used in a hardware description used to generate a TLM platform.

#### 3.4.4 Reference

This is not based upon the operation of a real device but is intended to be used for bring up and development of new virtual platforms.

Table 7. Configuration options (attributes) set for instance 'virtio\_mmio4'

Attribute	Value	Type	Expression
portAddress	0x10004000	uns32	
portSize	4096	uns32	
cbEnable	True	boolean	

### 3.5 Peripheral [[ovpworld.org/peripheral/trap/1.0](http://ovpworld.org/peripheral/trap/1.0)] instance: virtio\_mmio5

#### 3.5.1 Description

Open a port and allocate a region that is defined by parameters.

The region can be configured to act as standard memory or can report read/write accesses.

#### 3.5.2 Licensing

Open Source Apache 2.0

#### 3.5.3 Limitations

This peripheral cannot be used in a hardware description used to generate a TLM platform.

#### 3.5.4 Reference

This is not based upon the operation of a real device but is intended to be used for bring up and development of new virtual platforms.

Table 8. Configuration options (attributes) set for instance 'virtio\_mmio5'

Attribute	Value	Type	Expression
portAddress	0x10005000	uns32	
portSize	4096	uns32	
cbEnable	True	boolean	

### 3.6 Peripheral [[ovpworld.org/peripheral/trap/1.0](http://ovpworld.org/peripheral/trap/1.0)] instance: virtio\_mmio6

#### 3.6.1 Description

Open a port and allocate a region that is defined by parameters.

The region can be configured to act as standard memory or can report read/write accesses.

#### 3.6.2 Licensing

Open Source Apache 2.0

### 3.6.3 Limitations

This peripheral cannot be used in a hardware description used to generate a TLM platform.

### 3.6.4 Reference

This is not based upon the operation of a real device but is intended to be used for bring up and development of new virtual platforms.

Table 9. Configuration options (attributes) set for instance 'virtio\_mmio6'

Attribute	Value	Type	Expression
portAddress	0x10006000	uns32	
portSize	4096	uns32	
cbEnable	True	boolean	

## 3.7 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio\_mmio7

### 3.7.1 Description

Open a port and allocate a region that is defined by parameters.

The region can be configured to act as standard memory or can report read/write accesses.

### 3.7.2 Licensing

Open Source Apache 2.0

### 3.7.3 Limitations

This peripheral cannot be used in a hardware description used to generate a TLM platform.

### 3.7.4 Reference

This is not based upon the operation of a real device but is intended to be used for bring up and development of new virtual platforms.

Table 10. Configuration options (attributes) set for instance 'virtio\_mmio7'

Attribute	Value	Type	Expression
portAddress	0x10007000	uns32	
portSize	4096	uns32	
cbEnable	True	boolean	

## 3.8 Peripheral [ovpworld.org/peripheral/trap/1.0] instance: virtio\_mmio8

### 3.8.1 Description

Open a port and allocate a region that is defined by parameters.

The region can be configured to act as standard memory or can report read/write accesses.

### 3.8.2 Licensing

Open Source Apache 2.0

### 3.8.3 Limitations

This peripheral cannot be used in a hardware description used to generate a TLM platform.

### 3.8.4 Reference

This is not based upon the operation of a real device but is intended to be used for bring up and development of new virtual platforms.

Table 11. Configuration options (attributes) set for instance 'virtio\_mmio8'

Attribute	Value	Type	Expression
portAddress	0x10008000	uns32	
portSize	4096	uns32	
cbEnable	True	boolean	

## 3.9 Peripheral [riscv.ovpworld.org/peripheral/CLINT/1.0] instance: clint

### 3.9.1 Licensing

Open Source Apache 2.0

### 3.9.2 Description

SiFive-compatible Risc-V Core Local Interruptor (CLINT).

Use the num\_harts parameter to specify the number of harts supported (default 1).

For each supported hart there will be an MTimerInterruptN and MSWInterruptN net port, plus msipN and mtimecmpN registers implemented, where N is a value from 0..num\_harts-1.

There is also a single mtime register.

### 3.9.3 Reference

Various SiFive Core Complex manuals, e.g. SiFive U54 Core Complex Manual ([https://sifive.cdn.prismic.io/sifive/a07d1a9a-2cb8-4cf5-bb75-5351888ea2e1\\_u54\\_core\\_complex\\_manual\\_21G2.pdf](https://sifive.cdn.prismic.io/sifive/a07d1a9a-2cb8-4cf5-bb75-5351888ea2e1_u54_core_complex_manual_21G2.pdf))

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

Attribute	Value	Type	Expression
num_harts	1	Uns32	
clockMHz	25.0	double	

## 3.10 Peripheral [riscv.ovpworld.org/peripheral/PLIC/1.0] instance: plic

### 3.10.1 Licensing

Open Source Apache 2.0

### 3.10.2 Description

PLIC Interrupt Controller

Use parameters to configure specific implementation.

This is the generic version based on the RiscV Specification. Vendor-specific variations are available (e.g. Andes NCEPLIC100, SiFive PLIC).

### 3.10.3 Limitations

None

### 3.10.4 Reference

RISC-V Platform-Level Interrupt Controller Version 1.0, 09/2021 (<https://github.com/riscv/riscv-plic-spec/tree/832ec217197a59a375c5372c630e73fd954597bf>)

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

Attribute	Value	Type	Expression
num_sources	10	Uns32	
num_targets	2	Uns32	

## 3.11 Peripheral [[national.ovpworld.org/peripheral/16550/1.0](http://national.ovpworld.org/peripheral/16550/1.0)] instance: *uartTTY0*

### 3.11.1 Licensing

Open Source Apache 2.0

### 3.11.2 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.

### 3.11.3 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.

### 3.11.4 Reference

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

There are no configuration options set for this peripheral instance.

### **3.12 Peripheral [[riscv.ovpworld.org/peripheral/SmartLoaderRV64Linux/1.0](https://riscv.ovpworld.org/peripheral/SmartLoaderRV64Linux/1.0)] instance: smartLoader**

#### **3.12.1 Licensing**

Open Source Apache 2.0

#### **3.12.2 Description**

Pseudo-peripheral to insert boot code for a Riscv 64-bit Linux kernel boot. Loads simulated memory with a device tree blob file and boot code to set regs and jump to a Risc-v Linux Kernel.

#### **3.12.3 Limitations**

Only supports little endian

#### **3.12.4 Reference**

RISC-V Linux Kernel development

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

Attribute	Value	Type	Expression
bootaddr	0x80000000	address	
slbootaddr	0x1000	address	
membase	0x80000000	address	
memsize	0x08000000	address	

## 4.0 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 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. 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](http://www.OVPworld.org) site.

## 5.0 Getting Started with Imperas OVP Virtual Platforms

Virtual platforms are downloadable from the OVPworld website [OVPworld.org/downloads](http://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](http://OVPworld.org/downloads), or you can use one of the Imperas simulators ([imperas.com/products](http://imperas.com/products)) available commercially from Imperas.

## 6.0 Simulating Software

### 6.1 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](http://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.

### 6.2 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.

### 6.3 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](#).

#### ***6.4 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.

#### ***6.5 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.

#### ***6.6 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](#).

#### ***6.7 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](#).

### **7.0 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.



### **7.1 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](#).

### **7.2 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.

### **7.3 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.

### **7.4 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.

### **7.5 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](#).

### ***7.6 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.

## **8.0 Modifying the Platform / Module**

### ***8.1 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.

### ***8.2 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](http://imperas.com/products).

### ***8.3 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, 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 then run the simulator directly with the resultant shared object.

#### ***8.4 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.

#### ***8.5 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.

## 9.0 Available Virtual Platforms

Table 15. 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 16. 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 17. 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
ArmKernel	imperas.ovpworld.org
ArmKernelDual	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 18. 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
Or1kUlinux	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

#