



## OVP Guide to Using Processor Models

### Model specific information for RISC-V\_RVB32I

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	20231016.1
Filename	OVP_Model_Specific_Information_riscv_RVB32I.pdf
Created	7 December 2023
Status	OVP Standard Release

## Copyright Notice

Copyright (c) 2023 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).

# Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
1.1	Description	1
1.2	Licensing	1
1.3	Extensions	2
1.3.1	Extensions Enabled by Default	2
1.3.2	Enabling Other Extensions	2
1.4	General Features	3
1.4.1	mtvec CSR	3
1.4.2	Reset	3
1.4.3	NMI	3
1.4.4	WFI	4
1.4.5	time CSR	4
1.4.6	mcycle CSR	4
1.4.7	minstret CSR	4
1.4.8	mhpmcounter CSR	5
1.4.9	Unaligned Accesses	5
1.4.10	PMP	5
1.4.11	Time and Timers	5
1.5	Privileged Architecture	6
1.5.1	Legacy Version 1.10	6
1.5.2	Version 20190608	6
1.5.3	Version 20211203	6
1.5.4	Version 1.12	6
1.5.5	Version master	6
1.6	Unprivileged Architecture	7
1.6.1	Legacy Version 2.2	7
1.6.2	Version 20191213	7
1.7	Other Extensions	7
1.7.1	Zicsr	7
1.7.2	Zifencei	7
1.7.3	Zicbom	7
1.7.4	Zicbop	8
1.7.5	Zicboz	8
1.7.6	Smstateen	8
1.7.7	Sscofpmf	8
1.7.8	Zawrs	8
1.8	CLIC	8

1.9	Advanced Interrupt Architecture . . . . .	9
1.10	Interrupts . . . . .	9
1.11	Debug Mode (Sdext) . . . . .	9
1.11.1	Debug State Entry . . . . .	10
1.11.2	Debug State Exit . . . . .	11
1.11.3	Debug Registers . . . . .	11
1.11.4	Debug Mode Execution . . . . .	11
1.11.5	Debug Single Step . . . . .	12
1.11.6	Debug Event Priorities . . . . .	12
1.11.7	Debug Ports . . . . .	12
1.11.8	Debug Mode Versions . . . . .	12
1.11.9	Version 0.13.2-DRAFT . . . . .	12
1.11.10	Version 0.14.0-DRAFT . . . . .	13
1.11.11	Version 1.0.0-STABLE . . . . .	13
1.11.12	Version 1.0-STABLE . . . . .	13
1.12	Trigger Module (Sdtrig) . . . . .	13
1.12.1	Trigger Module Restrictions . . . . .	13
1.12.2	Trigger Module Parameters . . . . .	13
1.13	Debug Mask . . . . .	14
1.14	Integration Support . . . . .	14
1.14.1	Command “setPMA -attributes <attrs>-lo <addr>-hi <addr>” . . . . .	15
1.14.2	Command “getCSRIndex -name <name>” . . . . .	15
1.14.3	Command “listCSRs” . . . . .	15
1.14.4	CSR Register External Implementation . . . . .	15
1.14.5	External Stimulation of Illegal Instruction Trap . . . . .	16
1.15	Instruction Disassembly . . . . .	16
1.16	Limitations . . . . .	16
1.17	Verification . . . . .	17
1.18	References . . . . .	17
<b>2</b>	<b>Configuration</b>	<b>18</b>
2.1	Location . . . . .	18
2.2	GDB Path . . . . .	18
2.3	Semi-Host Library . . . . .	18
2.4	Processor Endian-ness . . . . .	18
2.5	QuantumLeap Support . . . . .	18
2.6	Processor ELF code . . . . .	18
<b>3</b>	<b>All Variants in this model</b>	<b>19</b>
<b>4</b>	<b>Bus Master Ports</b>	<b>21</b>
<b>5</b>	<b>Bus Slave Ports</b>	<b>22</b>
<b>6</b>	<b>Net Ports</b>	<b>23</b>
<b>7</b>	<b>FIFO Ports</b>	<b>24</b>
<b>8</b>	<b>Formal Parameters</b>	<b>25</b>

8.1	Parameters with enumerated types . . . . .	28
8.1.1	Parameter user_version . . . . .	28
8.1.2	Parameter priv_version . . . . .	29
8.1.3	Parameter compress_version . . . . .	29
8.1.4	Parameter debug_version . . . . .	29
8.1.5	Parameter rnmi_version . . . . .	29
8.1.6	Parameter Smepmp_version . . . . .	30
8.1.7	Parameter debug_mode . . . . .	30
8.1.8	Parameter chain_tval . . . . .	30
8.1.9	Parameter PMP_R0W1 . . . . .	30
8.2	Parameter values and limits . . . . .	30
<b>9</b>	<b>Execution Modes</b>	<b>34</b>
<b>10</b>	<b>Exceptions</b>	<b>35</b>
<b>11</b>	<b>Hierarchy of the model</b>	<b>36</b>
11.1	Level 1: Hart . . . . .	36
<b>12</b>	<b>Model Commands</b>	<b>37</b>
12.1	Level 1: Hart . . . . .	37
12.1.1	debugflags . . . . .	37
12.1.2	getCSRIndex . . . . .	37
12.1.3	isync . . . . .	37
12.1.4	itrace . . . . .	38
12.1.5	listCSRs . . . . .	38
12.1.5.1	Argument description . . . . .	38
12.1.6	setPMA . . . . .	38
<b>13</b>	<b>Registers</b>	<b>39</b>
13.1	Level 1: Hart . . . . .	39
13.1.1	Core . . . . .	39
13.1.2	Machine_Control_and_Status . . . . .	40
13.1.3	Integration_support . . . . .	44

# Chapter 1

## Overview

This document provides the details of an OVP Fast Processor Model variant.

OVP Fast Processor Models are written in C and provide a C API for use in C based platforms. The models also provide a native interface for use in SystemC TLM2 platforms.

The models are written using the OVP VMI API that provides a Virtual Machine Interface 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. Most 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 model families are validated using technology provided by the processor IP owners. There is a companion document (OVP Guide to Using Processor Models) which explains the general concepts of OVP Fast Processor Models and their use. It is downloadable from the OVPworld website documentation pages.

### 1.1 Description

RISC-V RVB32I 32-bit processor model

### 1.2 Licensing

This Model is released under the Open Source Apache 2.0

## 1.3 Extensions

### 1.3.1 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.

### 1.3.2 Enabling Other Extensions

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

misa bit 0: extension A (atomic instructions)

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 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.

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

implicit feature bit 1: extension B (bit manipulation extension)

implicit feature bit 10: extension K (cryptographic)

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.

## 1.4 General Features

### 1.4.1 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 `0xfffffd`. 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.

### 1.4.2 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.

### 1.4.3 NMI

An NMI input is implemented. To instead specify that NMI is not implemented, set parameter “`nmi_absent`” to `True`.

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`”.

The NMI input is level-sensitive. To instead specify that the NMI input is latched on the rising edge of the NMI signal, set parameter “`nmi_is_latched`” to True.

NMI interrupts are lower priority than Debug and Trigger Module events. To instead specify that NMI interrupts are higher priority, set parameter “`nmi_high_priority`” to True.

#### 1.4.4 WFI

WFI will halt the processor until an interrupt occurs. It can instead be configured as a NOP by setting parameter “`wfi_is_nop`” to True.

The nominal time limit for WFI instructions can be set by parameter “`TW_time_limit`”. In this variant, the time limit is 0 cycles.

Output signal “`core_wfi_mode`” indicates whether the processor is currently in WFI state.

Input signal “`restart_wfi`” will cause the processor to restart from WFI state when high.

Parameter “`wfi_resume_not_trap`” is 0 on this variant, meaning that pending wakeup events when WFI is executed will not prevent a trap occurring. If “`wfi_resume_not_trap`” is set to 1 then pending wakeup events when WFI is executed will cause the WFI to be treated as a NOP.

#### 1.4.5 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 section “Time and Timers” for more information.

#### 1.4.6 mcycle CSR

The “`mcycle`” CSR is implemented in this variant. Set parameter “`mcycle_undefined`” to True to instead specify that “`mcycle`” is unimplemented and accesses should cause Illegal Instruction traps.

#### 1.4.7 minstret CSR

The “`minstret`” CSR is implemented in this variant. Set parameter “`minstret_undefined`” to True to instead specify that “`minstret`” is unimplemented and accesses should cause Illegal Instruction traps.

### 1.4.8 mhpmmcounter CSR

The “mhpmmcounter” CSRs are implemented in this variant. Set parameter “mhpmmcounter\_undefined” to True to instead specify that “mhpmmcounter” CSRs are unimplemented and accesses should cause Illegal Instruction traps.

### 1.4.9 Unaligned Accesses

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

Address misaligned exceptions are higher priority than page fault or access fault exceptions on this variant. Set parameter “unaligned\_low\_pri” to “T” to specify that they are lower priority instead.

### 1.4.10 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.

Accesses to unimplemented PMP registers are write-ignored and read as zero on this variant. Set parameter “PMP\_undefined” to True to indicate that such accesses should cause Illegal Instruction exceptions instead.

### 1.4.11 Time and Timers

A RISC-V hart requires a time source to be available in any of the following cases:

1. The “time” CSR is implemented (“time\_undefined” is False);
2. The “Sstc” extension is present (“Sstc” is True);
3. The internal CLINT model is enabled (“CLINT\_address” is non-zero).

For cases 1 and 2, a 64-bit input port “mtime” is present. If this port is connected, it must be driven periodically by an external source with the current time value, which is visible in the “time” CSR and used for timer calculations by the “Sstc” extension. If the port is not connected, the value of time is internally derived with a period specified by the “mtime\_Hz” parameter (1e+06Hz by default).

For case 3, time is always internally derived and the “mtime” port is not present.

If the “time” CSR is implemented but the “Sstc” extension and the internal CLINT model are both absent, then it is also possible to implement the “time” CSR using a read callback on the CSR bus instead of using the “mtime” port: this may improve simulation performance if “time” increments at high frequency. See section “CSR Register External Implementation” for more information.

## 1.5 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.

### 1.5.1 Legacy Version 1.10

1.10 version of May 7 2017.

### 1.5.2 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);

### 1.5.3 Version 20211203

1.12 draft version of December 3 2021, 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.

### 1.5.4 Version 1.12

Official 1.12 version, identical to 20211203.

### 1.5.5 Version master

Unstable master version, currently identical to 1.12.

## 1.6 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.

### 1.6.1 Legacy Version 2.2

2.2 version of May 7 2017.

### 1.6.2 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.

## 1.7 Other Extensions

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

### 1.7.1 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.

### 1.7.2 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.

### 1.7.3 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.

#### 1.7.4 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).

#### 1.7.5 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”.

#### 1.7.6 Smstateen

Parameter “Smstateen” is 0 on this variant, meaning that state enable CSRs are undefined. if “Smstateen” is set to 1 then state enable CSRs are defined.

Within the state enable CSRs, only bit 1 (for Zfinx), bit 57 (for xcontext CSR access), bit 62 (for xenvcfg CSR access) and bit 63 (for lower-level state enable CSR access) are currently implemented.

#### 1.7.7 Sscofpmf

Parameter “Sscofpmf” is 0 on this variant, meaning that count overflow and mode-based filtering extension CSRs are undefined. if “Sscofpmf” is set to 1 then count overflow and mode-based filtering extension CSRs are defined.

Note that this model implements CSR state only for this extension; hardware performance counters themselves are implementation-specific and not implemented.

#### 1.7.8 Zawrs

Parameter “Zawrs” is 0 on this variant, meaning that wait-for-reservation-set instructions are not implemented. if “Zawrs” is set to 1 then wait-for-reservation-set instructions are implemented, in which case parameter “TW\_time\_limit” is used to specify the nominal cycle delay for wrs.nto, and parameter “STO\_time\_limit” is used to specify the nominal cycle delay for wrs.sto.

### 1.8 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.

## 1.9 Advanced Interrupt Architecture

The model can be configured to implement the Advanced Interrupt Architecture (AIA) interface using Boolean parameter “Smaia”; when True, the AIA interface is present as described in the RISC-V Advanced Interrupt Architecture specification, and further parameters are made available to configure other aspects of the interface. “Smaia” is False in this variant, indicating that the AIA interface is not implemented.

### 1.10 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 taken the value on the ID port is sampled and used to fill the Exception Code field in the relevant “xcause” CSR. For Machine External interrupts, the extra interrupt ID port is called “MExternalInterruptID”; for Supervisor External interrupts, the extra interrupt ID port is called “SEExternalInterruptID”.

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.

### 1.11 Debug Mode (Sdext)

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 four 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.
4. If set to value “inject”, then operations that would cause entry to Debug mode result in the processor continuing to execute from the current address in Debug mode. The harness should detect that Debug mode has been entered by monitoring the “DM” integration support register, and inject Debug-mode instructions one at a time using function `opProcessorSimulateInstruction`. Debug mode is exited by either an explicit write of False to the “DM” register or by execution of an injected “dret” instruction, as described by “Debug State Exit” below.

Note that parameter “debug\_mode” can be also be set to “none” to specify that Sdext is not implemented.

### 1.11.1 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`) - in this case, `dcsr.cause` will report a cause of trigger (2);
2. By writing a 1 then 0 to net “haltreq” (using `opNetWrite`) followed by simulation of at least one cycle (e.g. using `opProcessorSimulate`) - in this case, `dcsr.cause` will report a cause of haltreq (3);
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`) - in this case, `dcsr.cause` will report a cause of resethaltreq (5) or haltreq (3), depending on the value of parameter “no\_resethaltreq”;
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 this case, `dcsr.cause` will report a cause of ebreak (1);

5. By executing a single instruction when Debug mode entry for the current processor mode is enabled by `dcsr.step` - in this case, `dcsr.cause` will report a cause of step (4);
6. By a Trigger Module trigger, when that trigger is configured to enter Debug mode - in this case, `dcsr.cause` will report a cause of trigger (2).

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.

### 1.11.2 Debug State Exit

Exit from Debug mode can be performed in either 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.

### 1.11.3 Debug Registers

When Debug mode is enabled, registers “`dcsr`” and “`dpc`” are implemented as described in the specification. Registers “`dscratch0`” and “`dscratch1`” may also be implemented (see parameters below). Implemented registers 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.

Parameter “`dscratch0_undefined`” is used to specify whether the “`dscratch0`” CSR is undefined, in which case accesses to it trap to Machine mode. In this variant, “`dscratch0_undefined`” is 0.

Parameter “`dscratch1_undefined`” is used to specify whether the “`dscratch1`” CSR is undefined, in which case accesses to it trap to Machine mode. In this variant, “`dscratch1_undefined`” is 0.

### 1.11.4 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.

### 1.11.5 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 dcsr.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 dcsr.step is cleared.

### 1.11.6 Debug Event Priorities

The model supports three different models for determining which debug exception occurs when step, execute address, resethaltreq and haltreq events are all pending. These options are listed below, with highest-priority event first:

1. when parameter “debug\_priority”=“sxh”: step ->execute address ->resethaltreq ->haltreq;
2. when parameter “debug\_priority”=“shx”: step ->resethaltreq ->haltreq ->execute address;
3. when parameter “debug\_priority”=“hsx”: resethaltreq ->haltreq ->step ->execute address.

### 1.11.7 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).

### 1.11.8 Debug Mode Versions

Debug mode specification has been under active development. To enable simulation of hardware that may be based on an older version of the specification, the model implements behavior for a number of versions of the specification. The differing features of these are listed below, in chronological order.

### 1.11.9 Version 0.13.2-DRAFT

0.13.2-DRAFT version of March 22 2019.

### 1.11.10 Version 0.14.0-DRAFT

0.14.0-DRAFT version of November 6 2020.

### 1.11.11 Version 1.0.0-STABLE

1.0.0-STABLE version of February 9 2022.

### 1.11.12 Version 1.0-STABLE

1.0-STABLE version of December 28 2022, with these changes compared to version 1.0.0-STABLE:

- nmi is moved from etrigger to itrigger and is now subject to the mode bits in that trigger.

## 1.12 Trigger Module (Sdtrig)

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).

### 1.12.1 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.

### 1.12.2 Trigger Module Parameters

Parameter “trigger\_num” is used to specify the number of implemented triggers. In this variant, “trigger\_num” is 4. Set this parameter to 0 to specify that Sdtrig is not implemented.

Parameter “tinfo” is used to specify the value of the read-only “tinfo” register, which indicates the trigger types supported and also version information which controls the behavior of “mcontrol6”. In this variant, “tinfo” is 0x100807c.

Parameter “trigger\_match” is used to specify the legal “match” values for triggers of types 2 and 6. This parameter is a bitmask with 1 bits corresponding to legal values; for example, a “trigger\_match” of 0xd, means that triggers of types 0, 2 and 3 are supported. In this variant, “trigger\_match” is 0x333f.

Parameter “tdata2\_undefined” is used to specify whether the “tdata2” register is undefined, in which case reads of it trap to Machine mode. In this variant, “tdata2\_undefined” is 0.

Parameter “tdata3\_undefined” is used to specify whether the “tdata3” register is undefined, in which case reads of it trap to Machine mode. In this variant, “tdata3\_undefined” is 0.

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” bits in tdata1 are 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 6.

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 6.

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

## 1.13 Debug Mask

It is possible to enable model debug messages in various categories. This can be done statically using the “debugflags” 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.

## 1.14 Integration Support

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

### 1.14.1 Command “setPMA -attributes <attrs>-lo <addr>-hi <addr>”

This command allows PMA attributes to be set for the address range lo:hi. The required attributes are described by the “attrs” string, which can contain any combination of these characters:

“r”: allow read access

“w”: allow write access

“x”: allow execute access

“a”: disallow unaligned accesses

“A”: disallow RVMC\_USER1 accesses (often AMO and LR/SC)

“P”: disallow RVMC\_USER2 accesses (often push/pop)

“1”: allow 1-byte accesses

“2”: allow 2-byte accesses

“4”: allow 4-byte accesses

“8”: allow 8-byte accesses

<space>, “-”: ignored, use for formatting

The command may be used multiple times, in which case PMA attributes for later commands override those specified for earlier ones where ranges overlap. A common idiom is to deny all access to the entire memory range in the first command before adding back permissions for subregions with subsequent commands.

### 1.14.2 Command “getCSRIndex -name <name>”

This command returns the index number of a named CSR, or -1 if that CSR does not exist.

### 1.14.3 Command “listCSRs”

This command lists all implemented CSRs in index order.

### 1.14.4 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 a read callback at address 0xC010 on the CSR bus.

If both read and write callbacks are installed, or if a read callback is installed and the CSR is in the read-only address space, then the read callback will be used to provide the value for both true accesses and for trace and API register read (using opRegRead, etc). However, if only a read callback is installed and the CSR is in the CSR read/write address space then the callback will

be used for true register reads *\*only\**; in this case, the *\*model\** CSR implementation will be used for trace and API register read. This idiom allows values to be injected for volatile CSRs without changing fundamental model behavior.

An artifact net, “readcsr”, can also be used to override the value apparently read from a CSR without resorting to the CSR bus. When a CSR is read into a GPR that is not “x0”, this net is written with a value encoding the CSR number (in bits 11:0) and destination GPR number (in bits 20:16). To use this net:

1. Install a net monitor callback on “readcsr” using `opNetWriteMonitorAdd`;
2. When the callback is activated, extract the encoded CSR and GPR numbers;
3. If the CSR number corresponds to a CSR of interest, find the OP register corresponding to the GPR using `opProcessorRegByIndex`;
4. Use `opProcessorRegWrite` to modify the GPR value.

#### 1.14.5 External Stimulation of Illegal Instruction Trap

Artifact input net port “illegalinstr” allows Illegal Instruction traps to be raised externally. On a rising edge of the signal connected to this port, the hart will immediately take an Illegal Instruction trap with “xepc” set to the current program counter.

As a special case, if the hart is currently stalled by a WFI instruction (“wfi\_is\_nop” is False), it will be restarted and take either an Illegal Instruction or Virtual Instruction trap, based on the current processor mode and the governing TW bit.

### 1.15 Instruction Disassembly

This model implements a number of parameters to control instruction disassembly, as shown in trace output.

If parameter “use\_hw\_reg\_names” is True, instruction disassembly shows hardware names x0-x31. If “use\_hw\_reg\_names” is False, ABI names are shown instead.

If parameter “no\_pseudo\_inst” is True, instruction disassembly always shows true instructions. If “no\_pseudo\_inst” is False, pseudo-instructions are shown instead where applicable.

If parameter “show\_c\_prefix” is True, instruction disassembly of 16-bit instructions will include a compressed prefix (e.g. “c.” or “cm.”). If “show\_c\_prefix” is False, the compressed prefix will be omitted.

### 1.16 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.

## 1.17 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.

## 1.18 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 1.12, equivalent to 20211203)

RISC-V External Debug Support (RISC-V External Debug Support Version 1.0-STABLE as of commit 83483b1 of 21 August 2023 (this is subject to change))

# Chapter 2

## Configuration

### 2.1 Location

This model's VLNv is [riscv.ovpworld.org/processor/riscv/1.0](https://riscv.ovpworld.org/processor/riscv/1.0).

The model source is usually at:

`$IMPERAS_HOME/ImperasLib/source/riscv.ovpworld.org/processor/riscv/1.0`

The model binary is usually at:

`$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib/riscv.ovpworld.org/processor/riscv/1.0`

### 2.2 GDB Path

The default GDB for this model is: `$IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/riscv-none-embed-gdb`.

### 2.3 Semi-Host Library

The default semi-host library file is [riscv.ovpworld.org/semihosting/pk/1.0](https://riscv.ovpworld.org/semihosting/pk/1.0)

### 2.4 Processor Endian-ness

This is a LITTLE endian model.

### 2.5 QuantumLeap Support

This processor is qualified to run in a QuantumLeap enabled simulator.

### 2.6 Processor ELF code

The ELF code supported by this model is: 0xf3.

## Chapter 3

# All Variants in this model

This model has these variants

Variant	Description
RV32I	
RV32IM	
RV32IMC	
RV32IMCZ <sub>ce</sub>	
RV32IMAC	
RV32G	
RV32GC	
RV32GCZ <sub>finx</sub>	
RV32GCB	
RV32GCH	
RV32GCK	
RV32GCN	
RV32GCP	
RV32GCV	
RV32E	
RV32EC	
RV32EM	
RV64I	
RV64IM	
RV64IMC	
RV64IMCZ <sub>ce</sub>	
RV64IMAC	
RV64G	
RV64GC	
RV64GCZ <sub>finx</sub>	
RV64GCB	
RV64GCH	
RV64GCK	
RV64GCN	
RV64GCP	
RV64GCV	



RVB32I	(described in this document)
RVB32E	
RVB64I	
RVI20U32	
RVI20U64	
RVA20U64	
RVA20S64	
RVA22U64	
RVA22S64	

Table 3.1: All Variants in this model

## Chapter 4

# Bus Master Ports

This model has these bus master ports.

<b>Name</b>	min	max	Connect?	Description
INSTRUCTION	32	34	mandatory	Instruction bus
DATA	32	34	optional	Data bus

Table 4.1: Bus Master Ports

## Chapter 5

# Bus Slave Ports

This model has no bus slave ports.

## Chapter 6

# Net Ports

This model has these net ports.

Name	Type	Connect?	Description
reset	input	optional	Reset
reset_addr	input	optional	Externally-applied reset address
nmi	input	optional	NMI
nmi_cause	input	optional	Externally-applied NMI cause
nmi_addr	input	optional	Externally-applied NMI address
mtime	input	optional	External mtime source
MSWInterrupt	input	optional	Machine software interrupt
MTimerInterrupt	input	optional	Machine timer interrupt
MExternalInterrupt	input	optional	Machine external interrupt
irq_ack_o	output	optional	Interrupt acknowledge (pulse)
irq_id_o	output	optional	Acknowledged interrupt id (valid during irq_ack_o pulse)
sec_lvl_o	output	optional	Current privilege level
illegalinstr	input	optional	Artifact signal raising Illegal Instruction on rising edge
deferint	input	optional	Artifact signal causing interrupts to be held off when high
coverpoint	output	optional	Artifact port written with coverage point identifier
readcsr	output	optional	Artifact port written with CSR/GPR information when CSR is read
core_wfi_mode	output	optional	WFI is active
restart_wfi	input	optional	Artifact signal causing restart from WFI state when high

Table 6.1: Net Ports

## Chapter 7

# FIFO Ports

This model has no FIFO ports.

## Chapter 8

# Formal Parameters

Name	Type	Description
<b>Fundamental</b>		
user_version	Enumeration	Specify required User Architecture version
	2.2	User Architecture Version 2.2
	2.3	Deprecated and equivalent to 20191213
	20190305	Deprecated and equivalent to 20191213
	20191213	User Architecture Version 20191213
priv_version	Enumeration	Specify required Privileged Architecture version
	1.10	Privileged Architecture Version 1.10
	1.11	Privileged Architecture Version 1.11, equivalent to 20190608
	20190405	Deprecated and equivalent to 20190608
	20190608	Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11
	20211203	Privileged Architecture Version 20211203
	1.12	Privileged Architecture Version 1.12, equivalent to 20211203
	master	Privileged Architecture Master Branch as of commit 6bdeb58 (this is subject to change)
Smepmp_version	Enumeration	Specify required Smepmp Architecture version
	none	Smepmp not implemented
	0.9.5	Smepmp version 0.9.5 (deprecated and identical to 1.0)
	1.0	Smepmp version 1.0
numHarts	Uns32	Specify the number of hart contexts in a multiprocessor
enable_expanded	Boolean	Specify that 48-bit and 64-bit expanded instructions are supported
endianFixed	Boolean	Specify that data endianness is fixed (mstatus.{MBE,SBE,UBE} fields are read-only)
misa_MXL	Uns32	Override default value of misa.MXL
misa_Extensions	Uns32	Override default value of misa.Extensions
add_Extensions	String	Add extensions specified by letters to misa.Extensions (for example, specify “VD” to add V and D features)
sub_Extensions	String	Remove extensions specified by letters from misa.Extensions (for example, specify “VD” to remove V and D features)
misa_Extensions_mask	Uns32	Override mask of writable bits in misa.Extensions
add_Extensions_mask	String	Add extensions specified by letters to mask of writable bits in misa.Extensions (for example, specify “VD” to add V and D features)
sub_Extensions_mask	String	Remove extensions specified by letters from mask of writable bits in misa.Extensions (for example, specify “VD” to remove V and D features)
add_implicit_Extensions	String	Add extensions specified by letters to implicitly-present extensions not visible in misa.Extensions
sub_implicit_Extensions	String	Remove extensions specified by letters from implicitly-present extensions not visible in misa.Extensions
<b>Compressed Extension</b>		
compress_version	Enumeration	Specify required Compressed Architecture version

	legacy	Compressed Architecture absent or legacy version
	0.70.1	Compressed Architecture Version 0.70.1
	0.70.5	Compressed Architecture Version 0.70.5
	1.0.0-RC5.7	Compressed Architecture Version 1.0.0-RC5.7
	1.0	Compressed Architecture Version 1.0
<b>Debug_Extension</b>		
debug_version	Enumeration	Specify required Debug Architecture version
	0.13.2-DRAFT	RISC-V External Debug Support Version 0.13.2-DRAFT
	0.14.0-DRAFT	RISC-V External Debug Support Version 0.14.0-DRAFT
	1.0.0-STABLE	RISC-V External Debug Support Version 1.0.0-STABLE
	1.0-STABLE	RISC-V External Debug Support Version 1.0-STABLE as of commit 83483b1 of 21 August 2023 (this is subject to change)
debug_mode	Enumeration	Specify how Debug mode is implemented (value 'none' implies Sdext is not implemented)
	none	Debug mode not implemented
	vector	Debug mode implemented by execution at vector
	interrupt	Debug mode implemented by interrupt
	halt	Debug mode implemented by halt
	inject	Debug mode implemented using injected instructions
<b>Interrupts_Exceptions</b>		
rnmi_version	Enumeration	Specify required RNMI Architecture version
	none	RNMI not implemented
	0.2.1	RNMI version 0.2.1
	0.4_nmie1	RNMI version 0.4, except mnstatus.nmie=1 at reset
	0.4	RNMI version 0.4
mtvec_is_ro	Boolean	Specify whether mtvec CSR is read-only
tvec_align	Uns32	Specify hardware-enforced alignment of mtvec/stvec/utvec when Vectored interrupt mode enabled
ecode_mask	Uns64	Specify hardware-enforced mask of writable bits in xcause.ExceptionCode
ecode_nmi	Uns64	Specify xcause.ExceptionCode for NMI
nmi_absent	Boolean	Specify whether NMI input is absent
nmi_is_latched	Boolean	Specify whether NMI input is latched on rising edge (if False, it is level-sensitive)
nmi_high_priority	Boolean	Specify whether NMI input is higher priority than Debug and Trigger Module events
tval_zero	Boolean	Specify whether mtval/stval/utval are hard wired to zero
tval_zero_ebreak	Boolean	Specify whether mtval/stval/utval are set to zero by an ebreak
tval_ii_code	Boolean	Specify whether mtval/stval contain faulting instruction bits on illegal instruction exception
reset_address	Uns64	Override reset vector address
nmi_address	Uns64	Override NMI vector address
CLINT_address	Uns64	Specify base address of internal CLINT model (or 0 for no CLINT)
mtime_Hz	Double	Specify clock frequency of time CSR
local_int_num	Uns32	Specify number of supplemental local interrupts
unimp_int_mask	Uns64	Specify mask of unimplemented interrupts (e.g. 1<<9 indicates Supervisor external interrupt unimplemented)
force_mideleg	Uns64	Specify mask of interrupts always delegated to lower-priority execution level from Machine execution level
no_ideleg	Uns64	Specify mask of interrupts that cannot be delegated to lower-priority execution levels
no_e deleg	Uns64	Specify mask of exceptions that cannot be delegated to lower-priority execution levels
external_int_id	Boolean	Whether to add nets allowing External Interrupt ID codes to be forced
<b>Simulation_Artifact</b>		
leaf_hart_prefix	String	Specify string name prefix for harts in a cluster

use_hw_reg_names	Boolean	Specify whether to use hardware register names x0-x31 and f0-f31 instead of ABI register names
no_pseudo_inst	Boolean	Specify whether pseudo-instructions should not be reported in trace and disassembly
verbose	Boolean	Specify verbose output messages
traceVolatile	Boolean	Specify whether volatile registers (e.g. minstret) should be shown in change trace
wfi_restart	Boolean	Specify whether to automatically restart from WFI state when model is simulated
enable_CSR_bus	Boolean	Add artifact CSR bus port, allowing CSR registers to be externally implemented
CSR_remap	String	Comma-separated list of CSR number mappings, each of the form <csr-Name>=<number>
<b>Memory</b>		
unaligned_low_pri	Boolean	Specify whether address misaligned exceptions are lower priority than page or access fault exceptions
unaligned	Boolean	Specify whether the processor supports unaligned memory accesses
<b>Instruction_CSR_Behavior</b>		
wfi_is_nop	Boolean	Specify whether WFI should be treated as a NOP (if not, halt while waiting for interrupts)
wfi_resume_not_trap	Boolean	Specify whether pending wakeup events should cause WFI to be treated as a NOP instead of taking a trap
TW_time_limit	Uns32	Specify nominal cycle timeout for instructions controlled by mstatus.TW
counteren_mask	Uns32	Specify hardware-enforced mask of writable bits in mcounteren/scounteren registers
noinhibit_mask	Uns32	Specify hardware-enforced mask of always-zero bits in mcountinhibit register
mcycle_undefined	Boolean	Specify that the mcycle CSR is undefined
time_undefined	Boolean	Specify that the time CSR is undefined
minstret_undefined	Boolean	Specify that the minstret CSR is undefined
mhpmpcounter_undefined	Boolean	Specify that the mhpmpcounter CSRs are undefined
<b>CSR_Masks</b>		
mtvec_mask	Uns64	Specify hardware-enforced mask of writable bits in mtvec register
tdata1_mask	Uns64	Specify hardware-enforced mask of writable bits in Trigger Module tdata1 register
mip_mask	Uns64	Specify hardware-enforced mask of writable bits in mip register
envcfg_mask	Uns64	Specify hardware-enforced mask of writable bits in envcfg registers
mtvec_sext	Boolean	Specify whether mtvec is sign-extended from most-significant bit
<b>Trigger</b>		
tdata2_undefined	Boolean	Specify that the tdata2 CSR is undefined
tdata3_undefined	Boolean	Specify that the tdata3 CSR is undefined
tinfo_undefined	Boolean	Specify that the tinfo CSR is undefined
tcontrol_undefined	Boolean	Specify that the tcontrol CSR is undefined
mcontext_undefined	Boolean	Specify that the mcontext CSR is undefined
scontext_undefined	Boolean	Specify that the scontext CSR is undefined
mscontext_undefined	Boolean	Specify that the mscontext CSR is undefined (Debug Version 0.14.0 and later)
amo_trigger	Boolean	Specify whether AMO load/store operations activate triggers
no_hit	Boolean	Specify that tdata1.hit* bits are unimplemented
trigger_num	Uns32	Specify the number of implemented hardware triggers (0 implies Sdtrig is not implemented)
tinfo	Uns32	Override tinfo register (for all triggers)
trigger_match	Uns32	Specify legal “match” values for triggers of type 2 and 6 (bitmask)
mcontext_bits	Uns32	Specify the number of implemented bits in mcontext
mvalue_bits	Uns32	Specify the number of implemented bits in textra.mvalue (if zero, textra.mselect is tied to zero)



mcontrol_maskmax	Uns32	Specify mcontrol.maskmax value
chain_tval	Enumeration	Specify which trigger provides xtval when triggers are chained
	first	first matching trigger provides xtval
	last	last matching trigger provides xtval
	first_non_epc	first matching trigger provides xtval (prefer non-epc)
	last_non_epc	last matching trigger provides xtval (prefer non-epc)
<b>PMP Configuration</b>		
PMP_grain	Uns32	Specify PMP region granularity, G (0 =>4 bytes, 1 =>8 bytes, etc)
PMP_registers	Uns32	Specify the number of implemented PMP address registers
PMP_max_page	Uns32	Specify the maximum size of PMP region to map if non-zero (may improve performance; constrained to a power of two)
PMP_decompose	Boolean	Whether unaligned PMP accesses are decomposed into separate aligned accesses
PMP_undefined	Boolean	Whether accesses to unimplemented PMP registers are undefined (if True) or write ignored and zero (if False)
PMP_R0W1	Enumeration	Specify behavior of PMP RWX field on illegal write with R=0 and W=1
	RWX_00X	set R=0 and W=0, modify X
	RWX_11X	set R=1 and W=1, modify X
	RWX_PPX	preserve previous R and W, modify X
	RWX_PPP	preserve previous RWX
	RWX_000	set RWX=000
PMP_maskparams	Boolean	Enable parameters to change the read-only masks for PMP CSRs
PMP_initialparams	Boolean	Enable parameters to change the reset values for PMP CSRs
<b>Other Extensions</b>		
Smstateen	Boolean	Specify that Smstateen is implemented
Sscofpmf	Boolean	Specify that Sscofpmf is implemented
Zihintntnl	Boolean	Specify that Zihintntnl is implemented (instruction decode only, implemented as NOP)
Zicnd	Boolean	Specify that Zicnd is implemented
Zicsr	Boolean	Specify that Zicsr is implemented
Zifencei	Boolean	Specify that Zifencei is implemented
Zicbom	Boolean	Specify that Zicbom is implemented
Zicbop	Boolean	Specify that Zicbop is implemented
Zicboz	Boolean	Specify that Zicboz is implemented
Zawrs	Boolean	Specify that Zawrs is implemented
<b>CSR Defaults</b>		
mvendorid	Uns64	Override mvendorid register
marchid	Uns64	Override marchid register
mimpid	Uns64	Override mimpid register
mhartid	Uns64	Override mhartid register (or first mhartid of an incrementing sequence if this is an SMP variant)
mconfigptr	Uns64	Override mconfigptr register
mtvec	Uns64	Override mtvec register
mseccfg	Uns64	Override mseccfg register
<b>Fast Interrupt</b>		
CLICLEVELS	Uns32	Specify number of interrupt levels implemented by CLIC, or 0 if CLIC absent
<b>AIA Interrupts</b>		
Smaia	Boolean	Specify that Smaia CSRs are present

Table 8.1: Parameters that can be set in: Hart

## 8.1 Parameters with enumerated types

### 8.1.1 Parameter user\_version

Set to this value	Description
2.2	User Architecture Version 2.2
2.3	Deprecated and equivalent to 20191213
20190305	Deprecated and equivalent to 20191213
20191213	User Architecture Version 20191213

Table 8.2: Values for Parameter user\_version

### 8.1.2 Parameter priv\_version

Set to this value	Description
1.10	Privileged Architecture Version 1.10
1.11	Privileged Architecture Version 1.11, equivalent to 20190608
20190405	Deprecated and equivalent to 20190608
20190608	Privileged Architecture Version Ratified-IMFDQC-and-Priv-v1.11
20211203	Privileged Architecture Version 20211203
1.12	Privileged Architecture Version 1.12, equivalent to 20211203
master	Privileged Architecture Master Branch as of commit 6bdeb58 (this is subject to change)

Table 8.3: Values for Parameter priv\_version

### 8.1.3 Parameter compress\_version

Set to this value	Description
legacy	Compressed Architecture absent or legacy version
0.70.1	Compressed Architecture Version 0.70.1
0.70.5	Compressed Architecture Version 0.70.5
1.0.0-RC5.7	Compressed Architecture Version 1.0.0-RC5.7
1.0	Compressed Architecture Version 1.0

Table 8.4: Values for Parameter compress\_version

### 8.1.4 Parameter debug\_version

Set to this value	Description
0.13.2-DRAFT	RISC-V External Debug Support Version 0.13.2-DRAFT
0.14.0-DRAFT	RISC-V External Debug Support Version 0.14.0-DRAFT
1.0.0-STABLE	RISC-V External Debug Support Version 1.0.0-STABLE
1.0-STABLE	RISC-V External Debug Support Version 1.0-STABLE as of commit 83483b1 of 21 August 2023 (this is subject to change)

Table 8.5: Values for Parameter debug\_version

### 8.1.5 Parameter rnmi\_version

Set to this value	Description
none	RNMI not implemented
0.2.1	RNMI version 0.2.1
0.4_nmie1	RNMI version 0.4, except mnstatus.nmie=1 at reset
0.4	RNMI version 0.4

Table 8.6: Values for Parameter rnmi\_version

### 8.1.6 Parameter Smepmp\_version

Set to this value	Description
none	Smepmp not implemented
0.9.5	Smepmp version 0.9.5 (deprecated and identical to 1.0)
1.0	Smepmp version 1.0

Table 8.7: Values for Parameter Smepmp\_version

### 8.1.7 Parameter debug\_mode

Set to this value	Description
none	Debug mode not implemented
vector	Debug mode implemented by execution at vector
interrupt	Debug mode implemented by interrupt
halt	Debug mode implemented by halt
inject	Debug mode implemented using injected instructions

Table 8.8: Values for Parameter debug\_mode

### 8.1.8 Parameter chain\_tval

Set to this value	Description
first	first matching trigger provides xtval
last	last matching trigger provides xtval
first_non_epc	first matching trigger provides xtval (prefer non-epc)
last_non_epc	last matching trigger provides xtval (prefer non-epc)

Table 8.9: Values for Parameter chain\_tval

### 8.1.9 Parameter PMP\_R0W1

Set to this value	Description
RWX_00X	set R=0 and W=0, modify X
RWX_11X	set R=1 and W=1, modify X
RWX_PPX	preserve previous R and W, modify X
RWX_PPP	preserve previous RWX
RWX_000	set RWX=000

Table 8.10: Values for Parameter PMP\_R0W1

## 8.2 Parameter values and limits

These are the formal parameter limits and actual parameter values

Name	Min	Max	Default	Actual
<b>Fundamental</b>				
variant			RV32I	RVB32I
user_version			20191213	20191213
priv_version			1.12	1.12
Smepmp_version			none	none

numHarts	0	32	0	0
endian				none
enable_expanded			t	f
endianFixed			t	f
misa_MXL	1	2	1	1
misa_Extensions	0	67108863	256	0x100
add_Extensions				
sub_Extensions				
misa_Extensions_mask	0	67108863	256	0x100
add_Extensions_mask				
sub_Extensions_mask				
add_implicit_Extensions				
sub_implicit_Extensions				
<b>Compressed Extension</b>				
compress_version			1.0	1.0
<b>Debug Extension</b>				
debug_version			1.0-STABLE	1.0-STABLE
debug_mode			none	none
<b>Interrupts Exceptions</b>				
rnmi_version			none	none
mtvec_is_ro			t	f
tvec_align	0	65536	0	0
ecode_mask	0x0	0xffffffffffff	0x7ffffff	0x7ffffff
ecode_nmi	0x0	0xffffffffffff	0x0	0
nmi_absent			t	f
nmi_is_latched			t	f
nmi_high_priority			t	f
tval_zero			t	f
tval_zero_ebreak			t	f
tval_ii_code			t	t
reset_address	0x0	0xffffffffffff	0x0	0
nmi_address	0x0	0xffffffffffff	0x0	0
CLINT_address	0x0	0xffffffffffff	0x0	0
mtime_Hz	0.000000e+00	1.000000e+09	1.000000e+06	1.000000e+06
local_int_num	0	16	0	0
unimp_int_mask	0x0	0xffffffffffff	0x0	0
force_mideleg	0x0	0xffffffffffff	0x0	0
no_ideleg	0x0	0xffffffffffff	0x0	0
no_e deleg	0x0	0xffffffffffff	0x0	0
external_int_id			t	f
<b>Simulation Artifact</b>				
leaf_hart_prefix			hart	hart
use_hw_reg_names			t	f
no_pseudo_inst			t	f
verbose			t	f

traceVolatile			t	f
wfi_restart			t	f
enable_CSR_bus			t	f
CSR_remap				
<b>Memory</b>				
unaligned_low_pri			t	f
unaligned			t	f
<b>Instruction_CSR_Behavior</b>				
wfi_is_nop			t	f
wfi_resume_not_trap			t	f
TW_time_limit	0	4294967295	0	0
counteren_mask	0	4294967295	4294967295	0xffffffff
noinhibit_mask	0	4294967295	0	0
mcycle_undefined			t	f
time_undefined			t	f
minstret_undefined			t	f
mhpmcounter_undefined			t	f
<b>CSR Masks</b>				
mtvec_mask	0x0	0xffffffffffffff	0x0	0
tdata1_mask	0x0	0xffffffffffffff	0xffffffffffffff	0xffffffffffffff
mip_mask	0x0	0xffffffffffffff	0x337	0x337
envcfg_mask	0x0	0xffffffffffffff	0x0	0
mtvec_sext			t	f
<b>Trigger</b>				
tdata2_undefined			t	f
tdata3_undefined			t	f
tinfo_undefined			t	f
tcontrol_undefined			t	f
mcontext_undefined			t	f
scontext_undefined			t	f
mscontext_undefined			t	f
amo_trigger			t	f
no_hit			t	f
trigger_num	0	255	4	4
tinfo	0	16842751	16810108	0x100807c
trigger_match	1	65535	13119	0x333f
mcontext_bits	0	32	6	6
mvalue_bits	0	6	6	6
mcontrol_maskmax	0	63	63	63
chain_tval			first	first
<b>PMP Configuration</b>				
PMP_grain	0	29	0	0
PMP_registers	0	64	0	0
PMP_max_page	0	4294967295	0	0
PMP_decompose			t	f

PMP_undefined			t	f
PMP_R0W1			RWX_00X	RWX_00X
PMP_maskparams			t	f
PMP_initialparams			t	f
<b>Other Extensions</b>				
Smstateen			t	f
Sscofpmf			t	f
Zihintntl			t	f
Zicond			t	f
Zicsr			t	t
Zifencei			t	t
Zicbom			t	f
Zicbop			t	f
Zicboz			t	f
Zawrs			t	f
<b>CSR Defaults</b>				
mvendorid	0x0	0xffffffffffffff	0x0	0
marchid	0x0	0xffffffffffffff	0x0	0
mimpid	0x0	0xffffffffffffff	0x0	0
mhartid	0x0	0xffffffffffffff	0x0	0
mconfigptr	0x0	0xffffffffffffff	0x0	0
mtvec	0x0	0xffffffffffffff	0x0	0
mseccfg	0x0	0xffffffffffffff	0x0	0
<b>Fast Interrupt</b>				
CLICLEVELS	0	256	0	0
<b>AIA Interrupts</b>				
Smaia			t	f

Table 8.11: Parameter values and limits

## Chapter 9

# Execution Modes

Mode	Code	Description
Machine	3	Machine mode

Table 9.1: Modes implemented in: Hart

## Chapter 10

# Exceptions

Exception	Code	Description
InstructionAddressMisaligned	0	Fetch from unaligned address
InstructionAccessFault	1	No access permission for fetch
IllegalInstruction	2	Undecoded, unimplemented or disabled instruction
Breakpoint	3	EBREAK instruction executed
LoadAddressMisaligned	4	Load from unaligned address
LoadAccessFault	5	No access permission for load
StoreAMOAddressMisaligned	6	Store/atomic memory operation at unaligned address
StoreAMOAccessFault	7	No access permission for store/atomic memory operation
EnvironmentCallFromMMode	11	ECALL instruction executed in Machine mode
MSWInterrupt	67	Machine software interrupt
MTimerInterrupt	71	Machine timer interrupt
MExternalInterrupt	75	Machine external interrupt
GenericNMI	4294967295	Generic NMI

Table 10.1: Exceptions implemented in: Hart



# Chapter 11

## Hierarchy of the model

A CPU core may be configured to instance many processors of a Symmetrical Multi Processor (SMP). A CPU core may also have sub elements within a processor, for example hardware threading blocks.

OVP processor models can be written to include SMP blocks and to have many levels of hierarchy. Some OVP CPU models may have a fixed hierarchy, and some may be configured by settings in a configuration register. Please see the register definitions of this model.

This model documentation shows the settings and hierarchy of the default settings for this model variant.

### 11.1 Level 1: Hart

This level in the model hierarchy has 6 commands.

This level in the model hierarchy has 3 register groups:

Group name	Registers
Core	33
Machine_Control_and_Status	260
Integration_support	2

Table 11.1: Register groups

This level in the model hierarchy has no children.

# Chapter 12

## Model Commands

A Processor model can implement one or more **Model Commands** available to be invoked from the simulator command line, from the OP API or from the Imperas Multiprocessor Debugger.

### 12.1 Level 1: Hart

#### 12.1.1 debugflags

show or modify the processor debug flags

Argument	Type	Description
-get	Boolean	print current processor flags value
-mask	Boolean	print valid debug flag bits
-set	Int32	new processor flags (only flags 0x00000006 can be modified)

Table 12.1: debugflags command arguments

#### 12.1.2 getCSRIndex

Return index for a named CSR (or -1 if no matching CSR)

Argument	Type	Description
-name	String	CSR name

Table 12.2: getCSRIndex command arguments

#### 12.1.3 isync

specify instruction address range for synchronous execution

Argument	Type	Description
-addresshi	Uns64	end address of synchronous execution range
-addresslo	Uns64	start address of synchronous execution range

Table 12.3: isync command arguments

### 12.1.4 itrace

enable or disable instruction tracing

Argument	Type	Description
-access	String	show memory accesses by this instruction. Argument can be any combination of X (execute), A (load or store access) and S (system)
-after	Uns64	apply after this many instructions
-enable	Boolean	enable instruction tracing
-full	Boolean	turn on all trace features
-instructioncount	Boolean	include the instruction number in each trace
-memory	String	(Alias for access). show memory accesses by this instruction. Argument can be any combination of X (execute), A (load or store access) and S (system)
-mode	Boolean	show processor mode changes
-off	Boolean	disable instruction tracing
-on	Boolean	enable instruction tracing
-processorname	Boolean	Include processor name in all trace lines
-registerchange	Boolean	show registers changed by this instruction
-registers	Boolean	show registers after each trace

Table 12.4: itrace command arguments

### 12.1.5 listCSRs

#### 12.1.5.1 Argument description

List all CSRs in index order

### 12.1.6 setPMA

Set PMA region permissions and legal access sizes

Argument	Type	Description
-attributes	String	region attributes (string containing r, w, x, a, A, P, 1, 2, 4 or 8)
-hi	Uns64	high address
-lo	Uns64	low address

Table 12.5: setPMA command arguments

# Chapter 13

## Registers

### 13.1 Level 1: Hart

#### 13.1.1 Core

Registers at level:1, type:Hart group:Core

Name	Bits	Initial-Hex	RW	Description
zero	32	0	r-	
ra	32	0	rw	
sp	32	0	rw	stack pointer
gp	32	0	rw	
tp	32	0	rw	
t0	32	0	rw	
t1	32	0	rw	
t2	32	0	rw	
s0	32	0	rw	
s1	32	0	rw	
a0	32	0	rw	
a1	32	0	rw	
a2	32	0	rw	
a3	32	0	rw	
a4	32	0	rw	
a5	32	0	rw	
a6	32	0	rw	
a7	32	0	rw	
s2	32	0	rw	
s3	32	0	rw	
s4	32	0	rw	
s5	32	0	rw	
s6	32	0	rw	
s7	32	0	rw	
s8	32	0	rw	
s9	32	0	rw	
s10	32	0	rw	
s11	32	0	rw	
t3	32	0	rw	
t4	32	0	rw	
t5	32	0	rw	
t6	32	0	rw	
pc	32	0	rw	program counter

Table 13.1: Registers at level 1, type:Hart group:Core

### 13.1.2 Machine\_Control\_and\_Status

Registers at level:1, type:Hart group:Machine\_Control\_and\_Status

Name	Bits	Initial-Hex	RW	Description
mstatus	32	1800	rw	Machine Status
misa	32	40000100	rw	ISA and Extensions
mie	32	0	rw	Machine Interrupt Enable
mtvec	32	0	rw	Machine Trap-Vector Base-Address
mstatush	32	0	rw	Machine Status High
mcountinhibit	32	0	rw	Machine Counter Inhibit
mhpmevent3	32	0	rw	Machine Performance Monitor Event Select 3
mhpmevent4	32	0	rw	Machine Performance Monitor Event Select 4
mhpmevent5	32	0	rw	Machine Performance Monitor Event Select 5
mhpmevent6	32	0	rw	Machine Performance Monitor Event Select 6
mhpmevent7	32	0	rw	Machine Performance Monitor Event Select 7
mhpmevent8	32	0	rw	Machine Performance Monitor Event Select 8
mhpmevent9	32	0	rw	Machine Performance Monitor Event Select 9
mhpmevent10	32	0	rw	Machine Performance Monitor Event Select 10
mhpmevent11	32	0	rw	Machine Performance Monitor Event Select 11
mhpmevent12	32	0	rw	Machine Performance Monitor Event Select 12
mhpmevent13	32	0	rw	Machine Performance Monitor Event Select 13
mhpmevent14	32	0	rw	Machine Performance Monitor Event Select 14
mhpmevent15	32	0	rw	Machine Performance Monitor Event Select 15
mhpmevent16	32	0	rw	Machine Performance Monitor Event Select 16
mhpmevent17	32	0	rw	Machine Performance Monitor Event Select 17
mhpmevent18	32	0	rw	Machine Performance Monitor Event Select 18
mhpmevent19	32	0	rw	Machine Performance Monitor Event Select 19
mhpmevent20	32	0	rw	Machine Performance Monitor Event Select 20
mhpmevent21	32	0	rw	Machine Performance Monitor Event Select 21
mhpmevent22	32	0	rw	Machine Performance Monitor Event Select 22
mhpmevent23	32	0	rw	Machine Performance Monitor Event Select 23
mhpmevent24	32	0	rw	Machine Performance Monitor Event Select 24
mhpmevent25	32	0	rw	Machine Performance Monitor Event Select 25
mhpmevent26	32	0	rw	Machine Performance Monitor Event Select 26
mhpmevent27	32	0	rw	Machine Performance Monitor Event Select 27
mhpmevent28	32	0	rw	Machine Performance Monitor Event Select 28
mhpmevent29	32	0	rw	Machine Performance Monitor Event Select 29
mhpmevent30	32	0	rw	Machine Performance Monitor Event Select 30
mhpmevent31	32	0	rw	Machine Performance Monitor Event Select 31
mscratch	32	0	rw	Machine Scratch
mepc	32	0	rw	Machine Exception Program Counter
mcause	32	0	rw	Machine Cause
mtval	32	0	rw	Machine Trap Value
mip	32	0	rw	Machine Interrupt Pending
pmpcfg0	32	0	rw	Physical Memory Protection Configuration 0
pmpcfg1	32	0	rw	Physical Memory Protection Configuration 1
pmpcfg2	32	0	rw	Physical Memory Protection Configuration 2
pmpcfg3	32	0	rw	Physical Memory Protection Configuration 3
pmpcfg4	32	0	rw	Physical Memory Protection Configuration 4
pmpcfg5	32	0	rw	Physical Memory Protection Configuration 5
pmpcfg6	32	0	rw	Physical Memory Protection Configuration 6
pmpcfg7	32	0	rw	Physical Memory Protection Configuration 7

pmpcfg8	32	0	rw	Physical Memory Protection Configuration 8
pmpcfg9	32	0	rw	Physical Memory Protection Configuration 9
pmpcfg10	32	0	rw	Physical Memory Protection Configuration 10
pmpcfg11	32	0	rw	Physical Memory Protection Configuration 11
pmpcfg12	32	0	rw	Physical Memory Protection Configuration 12
pmpcfg13	32	0	rw	Physical Memory Protection Configuration 13
pmpcfg14	32	0	rw	Physical Memory Protection Configuration 14
pmpcfg15	32	0	rw	Physical Memory Protection Configuration 15
pmpaddr0	32	0	rw	Physical Memory Protection Address 0
pmpaddr1	32	0	rw	Physical Memory Protection Address 1
pmpaddr2	32	0	rw	Physical Memory Protection Address 2
pmpaddr3	32	0	rw	Physical Memory Protection Address 3
pmpaddr4	32	0	rw	Physical Memory Protection Address 4
pmpaddr5	32	0	rw	Physical Memory Protection Address 5
pmpaddr6	32	0	rw	Physical Memory Protection Address 6
pmpaddr7	32	0	rw	Physical Memory Protection Address 7
pmpaddr8	32	0	rw	Physical Memory Protection Address 8
pmpaddr9	32	0	rw	Physical Memory Protection Address 9
pmpaddr10	32	0	rw	Physical Memory Protection Address 10
pmpaddr11	32	0	rw	Physical Memory Protection Address 11
pmpaddr12	32	0	rw	Physical Memory Protection Address 12
pmpaddr13	32	0	rw	Physical Memory Protection Address 13
pmpaddr14	32	0	rw	Physical Memory Protection Address 14
pmpaddr15	32	0	rw	Physical Memory Protection Address 15
pmpaddr16	32	0	rw	Physical Memory Protection Address 16
pmpaddr17	32	0	rw	Physical Memory Protection Address 17
pmpaddr18	32	0	rw	Physical Memory Protection Address 18
pmpaddr19	32	0	rw	Physical Memory Protection Address 19
pmpaddr20	32	0	rw	Physical Memory Protection Address 20
pmpaddr21	32	0	rw	Physical Memory Protection Address 21
pmpaddr22	32	0	rw	Physical Memory Protection Address 22
pmpaddr23	32	0	rw	Physical Memory Protection Address 23
pmpaddr24	32	0	rw	Physical Memory Protection Address 24
pmpaddr25	32	0	rw	Physical Memory Protection Address 25
pmpaddr26	32	0	rw	Physical Memory Protection Address 26
pmpaddr27	32	0	rw	Physical Memory Protection Address 27
pmpaddr28	32	0	rw	Physical Memory Protection Address 28
pmpaddr29	32	0	rw	Physical Memory Protection Address 29
pmpaddr30	32	0	rw	Physical Memory Protection Address 30
pmpaddr31	32	0	rw	Physical Memory Protection Address 31
pmpaddr32	32	0	rw	Physical Memory Protection Address 32
pmpaddr33	32	0	rw	Physical Memory Protection Address 33
pmpaddr34	32	0	rw	Physical Memory Protection Address 34
pmpaddr35	32	0	rw	Physical Memory Protection Address 35
pmpaddr36	32	0	rw	Physical Memory Protection Address 36
pmpaddr37	32	0	rw	Physical Memory Protection Address 37
pmpaddr38	32	0	rw	Physical Memory Protection Address 38
pmpaddr39	32	0	rw	Physical Memory Protection Address 39
pmpaddr40	32	0	rw	Physical Memory Protection Address 40
pmpaddr41	32	0	rw	Physical Memory Protection Address 41
pmpaddr42	32	0	rw	Physical Memory Protection Address 42
pmpaddr43	32	0	rw	Physical Memory Protection Address 43
pmpaddr44	32	0	rw	Physical Memory Protection Address 44
pmpaddr45	32	0	rw	Physical Memory Protection Address 45
pmpaddr46	32	0	rw	Physical Memory Protection Address 46
pmpaddr47	32	0	rw	Physical Memory Protection Address 47

pmpaddr48	32	0	rw	Physical Memory Protection Address 48
pmpaddr49	32	0	rw	Physical Memory Protection Address 49
pmpaddr50	32	0	rw	Physical Memory Protection Address 50
pmpaddr51	32	0	rw	Physical Memory Protection Address 51
pmpaddr52	32	0	rw	Physical Memory Protection Address 52
pmpaddr53	32	0	rw	Physical Memory Protection Address 53
pmpaddr54	32	0	rw	Physical Memory Protection Address 54
pmpaddr55	32	0	rw	Physical Memory Protection Address 55
pmpaddr56	32	0	rw	Physical Memory Protection Address 56
pmpaddr57	32	0	rw	Physical Memory Protection Address 57
pmpaddr58	32	0	rw	Physical Memory Protection Address 58
pmpaddr59	32	0	rw	Physical Memory Protection Address 59
pmpaddr60	32	0	rw	Physical Memory Protection Address 60
pmpaddr61	32	0	rw	Physical Memory Protection Address 61
pmpaddr62	32	0	rw	Physical Memory Protection Address 62
pmpaddr63	32	0	rw	Physical Memory Protection Address 63
scontext	32	0	rw	Trigger Supervisor Context
tselect	32	0	rw	Trigger Register Select
tdata1	32	f0000000	rw	Trigger Data 1
tdata2	32	0	rw	Trigger Data 2
tdata3	32	0	rw	Trigger Data 3
tinfo	32	100807c	rw	Trigger Info
tcontrol	32	0	rw	Trigger Control
mcontext	32	0	rw	Trigger Machine Context
mscontext	32	0	rw	Trigger Machine Context Alias
mcycle	32	0	rw	Machine Cycle Counter
minstret	32	0	rw	Machine Instructions Retired
mhpmcounter3	32	0	rw	Machine Performance Monitor Counter 3
mhpmcounter4	32	0	rw	Machine Performance Monitor Counter 4
mhpmcounter5	32	0	rw	Machine Performance Monitor Counter 5
mhpmcounter6	32	0	rw	Machine Performance Monitor Counter 6
mhpmcounter7	32	0	rw	Machine Performance Monitor Counter 7
mhpmcounter8	32	0	rw	Machine Performance Monitor Counter 8
mhpmcounter9	32	0	rw	Machine Performance Monitor Counter 9
mhpmcounter10	32	0	rw	Machine Performance Monitor Counter 10
mhpmcounter11	32	0	rw	Machine Performance Monitor Counter 11
mhpmcounter12	32	0	rw	Machine Performance Monitor Counter 12
mhpmcounter13	32	0	rw	Machine Performance Monitor Counter 13
mhpmcounter14	32	0	rw	Machine Performance Monitor Counter 14
mhpmcounter15	32	0	rw	Machine Performance Monitor Counter 15
mhpmcounter16	32	0	rw	Machine Performance Monitor Counter 16
mhpmcounter17	32	0	rw	Machine Performance Monitor Counter 17
mhpmcounter18	32	0	rw	Machine Performance Monitor Counter 18
mhpmcounter19	32	0	rw	Machine Performance Monitor Counter 19
mhpmcounter20	32	0	rw	Machine Performance Monitor Counter 20
mhpmcounter21	32	0	rw	Machine Performance Monitor Counter 21
mhpmcounter22	32	0	rw	Machine Performance Monitor Counter 22
mhpmcounter23	32	0	rw	Machine Performance Monitor Counter 23
mhpmcounter24	32	0	rw	Machine Performance Monitor Counter 24
mhpmcounter25	32	0	rw	Machine Performance Monitor Counter 25
mhpmcounter26	32	0	rw	Machine Performance Monitor Counter 26
mhpmcounter27	32	0	rw	Machine Performance Monitor Counter 27
mhpmcounter28	32	0	rw	Machine Performance Monitor Counter 28
mhpmcounter29	32	0	rw	Machine Performance Monitor Counter 29
mhpmcounter30	32	0	rw	Machine Performance Monitor Counter 30
mhpmcounter31	32	0	rw	Machine Performance Monitor Counter 31

mcycleh	32	0	rw	Machine Cycle Counter High
minstreth	32	0	rw	Machine Instructions Retired High
mhpmpcounterh3	32	0	rw	Machine Performance Monitor Counter High 3
mhpmpcounterh4	32	0	rw	Machine Performance Monitor Counter High 4
mhpmpcounterh5	32	0	rw	Machine Performance Monitor Counter High 5
mhpmpcounterh6	32	0	rw	Machine Performance Monitor Counter High 6
mhpmpcounterh7	32	0	rw	Machine Performance Monitor Counter High 7
mhpmpcounterh8	32	0	rw	Machine Performance Monitor Counter High 8
mhpmpcounterh9	32	0	rw	Machine Performance Monitor Counter High 9
mhpmpcounterh10	32	0	rw	Machine Performance Monitor Counter High 10
mhpmpcounterh11	32	0	rw	Machine Performance Monitor Counter High 11
mhpmpcounterh12	32	0	rw	Machine Performance Monitor Counter High 12
mhpmpcounterh13	32	0	rw	Machine Performance Monitor Counter High 13
mhpmpcounterh14	32	0	rw	Machine Performance Monitor Counter High 14
mhpmpcounterh15	32	0	rw	Machine Performance Monitor Counter High 15
mhpmpcounterh16	32	0	rw	Machine Performance Monitor Counter High 16
mhpmpcounterh17	32	0	rw	Machine Performance Monitor Counter High 17
mhpmpcounterh18	32	0	rw	Machine Performance Monitor Counter High 18
mhpmpcounterh19	32	0	rw	Machine Performance Monitor Counter High 19
mhpmpcounterh20	32	0	rw	Machine Performance Monitor Counter High 20
mhpmpcounterh21	32	0	rw	Machine Performance Monitor Counter High 21
mhpmpcounterh22	32	0	rw	Machine Performance Monitor Counter High 22
mhpmpcounterh23	32	0	rw	Machine Performance Monitor Counter High 23
mhpmpcounterh24	32	0	rw	Machine Performance Monitor Counter High 24
mhpmpcounterh25	32	0	rw	Machine Performance Monitor Counter High 25
mhpmpcounterh26	32	0	rw	Machine Performance Monitor Counter High 26
mhpmpcounterh27	32	0	rw	Machine Performance Monitor Counter High 27
mhpmpcounterh28	32	0	rw	Machine Performance Monitor Counter High 28
mhpmpcounterh29	32	0	rw	Machine Performance Monitor Counter High 29
mhpmpcounterh30	32	0	rw	Machine Performance Monitor Counter High 30
mhpmpcounterh31	32	0	rw	Machine Performance Monitor Counter High 31
cycle	32	0	r-	Cycle Counter
time	32	0	r-	Timer
instret	32	0	r-	Instructions Retired
hpmcounter3	32	0	r-	Performance Monitor Counter 3
hpmcounter4	32	0	r-	Performance Monitor Counter 4
hpmcounter5	32	0	r-	Performance Monitor Counter 5
hpmcounter6	32	0	r-	Performance Monitor Counter 6
hpmcounter7	32	0	r-	Performance Monitor Counter 7
hpmcounter8	32	0	r-	Performance Monitor Counter 8
hpmcounter9	32	0	r-	Performance Monitor Counter 9
hpmcounter10	32	0	r-	Performance Monitor Counter 10
hpmcounter11	32	0	r-	Performance Monitor Counter 11
hpmcounter12	32	0	r-	Performance Monitor Counter 12
hpmcounter13	32	0	r-	Performance Monitor Counter 13
hpmcounter14	32	0	r-	Performance Monitor Counter 14
hpmcounter15	32	0	r-	Performance Monitor Counter 15
hpmcounter16	32	0	r-	Performance Monitor Counter 16
hpmcounter17	32	0	r-	Performance Monitor Counter 17
hpmcounter18	32	0	r-	Performance Monitor Counter 18
hpmcounter19	32	0	r-	Performance Monitor Counter 19
hpmcounter20	32	0	r-	Performance Monitor Counter 20
hpmcounter21	32	0	r-	Performance Monitor Counter 21
hpmcounter22	32	0	r-	Performance Monitor Counter 22
hpmcounter23	32	0	r-	Performance Monitor Counter 23
hpmcounter24	32	0	r-	Performance Monitor Counter 24



hpmcounter25	32	0	r-	Performance Monitor Counter 25
hpmcounter26	32	0	r-	Performance Monitor Counter 26
hpmcounter27	32	0	r-	Performance Monitor Counter 27
hpmcounter28	32	0	r-	Performance Monitor Counter 28
hpmcounter29	32	0	r-	Performance Monitor Counter 29
hpmcounter30	32	0	r-	Performance Monitor Counter 30
hpmcounter31	32	0	r-	Performance Monitor Counter 31
cycleh	32	0	r-	Cycle Counter High
timeh	32	0	r-	Timer High
instreth	32	0	r-	Instructions Retired High
hpmcounterh3	32	0	r-	Performance Monitor High 3
hpmcounterh4	32	0	r-	Performance Monitor High 4
hpmcounterh5	32	0	r-	Performance Monitor High 5
hpmcounterh6	32	0	r-	Performance Monitor High 6
hpmcounterh7	32	0	r-	Performance Monitor High 7
hpmcounterh8	32	0	r-	Performance Monitor High 8
hpmcounterh9	32	0	r-	Performance Monitor High 9
hpmcounterh10	32	0	r-	Performance Monitor High 10
hpmcounterh11	32	0	r-	Performance Monitor High 11
hpmcounterh12	32	0	r-	Performance Monitor High 12
hpmcounterh13	32	0	r-	Performance Monitor High 13
hpmcounterh14	32	0	r-	Performance Monitor High 14
hpmcounterh15	32	0	r-	Performance Monitor High 15
hpmcounterh16	32	0	r-	Performance Monitor High 16
hpmcounterh17	32	0	r-	Performance Monitor High 17
hpmcounterh18	32	0	r-	Performance Monitor High 18
hpmcounterh19	32	0	r-	Performance Monitor High 19
hpmcounterh20	32	0	r-	Performance Monitor High 20
hpmcounterh21	32	0	r-	Performance Monitor High 21
hpmcounterh22	32	0	r-	Performance Monitor High 22
hpmcounterh23	32	0	r-	Performance Monitor High 23
hpmcounterh24	32	0	r-	Performance Monitor High 24
hpmcounterh25	32	0	r-	Performance Monitor High 25
hpmcounterh26	32	0	r-	Performance Monitor High 26
hpmcounterh27	32	0	r-	Performance Monitor High 27
hpmcounterh28	32	0	r-	Performance Monitor High 28
hpmcounterh29	32	0	r-	Performance Monitor High 29
hpmcounterh30	32	0	r-	Performance Monitor High 30
hpmcounterh31	32	0	r-	Performance Monitor High 31
mvendorid	32	0	r-	Vendor ID
marchid	32	0	r-	Architecture ID
mimpid	32	0	r-	Implementation ID
mhartid	32	0	r-	Hardware Thread ID
mconfigptr	32	0	r-	Configuration Data Structure

Table 13.2: Registers at level 1, type:Hart group:Machine\_Control\_and\_Status

### 13.1.3 Integration support

Registers at level:1, type:Hart group:Integration\_support

Name	Bits	Initial-Hex	RW	Description
commercial	8	0	r-	Commercial feature in use
ASYNCPE	8	0	r-	Asynchronous Event Pending & Enabled

Table 13.3: Registers at level 1, type:Hart group:Integration\_support