RISC-V Platform Specification

RISC-V Platform Horizontal Subcommittee (RISC-V Platform HSC)

Version 0.3-draft, December 2021: This document is in Development state. Change should be expected.

Table of Contents

Preamble

This document is in the Development state

Assume everything can change. This draft specification will change before being accepted as standard, so implementations made to this draft specification will likely not conform to the future standard.

Copyright and License Information

This RISC-V Profile and Platform Specification (P2S) is

Change Log

version 0.3-draft

- 2021-12-13:
  - Restructure document into OS-A Common, OS-A Embedded and OS-A Server

version 0.2-draft

- 2021-09-01:
  - Draft version for internal reviews

version 0.1-draft

- 2020-10-07:
  - Initial changes for structure and future maintenance.
  - Break content down into include files; more structure, but easier to make changes down the line.

Terminology

TERM	DESCRIPTION
SBI	Supervisor Binary Interface [6]
UEFI	Unified Extensible Firmware Interface [1]
ACPI	Advanced Configuration and Power Interface [16]
APEI	ACPI Platform Error Interfaces [17]
SMBIOS	System Management Basic I/O System [18]
DTS	Devicetree source file [2]
DTB	Devicetree binary [2]
RVA22	RISC-V Application 2022 [11]
EE	Execution Environment
OSPM	Operating System Power Management
RVA22U64	RISC-V 2022 user-mode profile [11]
RVA22S64	RISC-V 2022 supervisor-mode profile [11]
RAS	Reliability, Availability, and Serviceability
CLINT	Legacy Core-Local Interrupt Controller
ACLINT	Advanced Core-Local Interrupt Controller [9]
PLIC	Legacy Platform-Level Interrupt Controller [7]
APLIC	Advanced Platform-Level Interrupt Controller [10]
AIA	Advanced Interrupt Architecture [10]
IMSIC	Incomning MSI Controller [10]
L1D	L1 Data cache
LL	Last level cache
DTLB	DATA TLB cache
PCIe	PCI Express
ECAM	Enhanced Configuration Access Mechanism
BAR	Base Address Register
AER	Advanced Error Reporting
CRS	Configuration Request Retry Status
TLP	Transaction Layer Packet
RCiEP	Root Complex Integrated Endpoint
RCEC	Root Complex Event Collector
PME	Power Management Event
MSI	Message Signaled Interrupts
MSI-X	Enhanced Message Signaled Interrupts

TERM	DESCRIPTION
INTx	PCIe Legacy Interrupts
PMA	Physical Memory Attributes
PRT	PCI Routing Table
EBBR	Embedded Base Boot Requirements [15]

Chapter 1. Introduction

The platform specification defines a set of platforms that specify requirements for interoperability between software and hardware. The Platform Policy [23] defines the various terms used in this platform specification. The platform policy also provides the needed detail regarding the scope, coverage, naming, versioning, structure, life cycle and compatibility claims for the platform specification. It is recommended that readers get familiar with the platform policy while reading this specification. All the requirements in this specification are MANDATORY unless specifically called out in the relevant sections. Any hardware platform seeking compatibility with the platform specification has to be self certified by the platform compatibility test suite (PCT). More details about the PCT are available in the platform policy specification.

The platform specification currently defines two platforms as shown below. Additional platforms are expected to be defined in the future for industry specific target market verticals like “mobile”, “edge computing”, “machine-learning” "desktop", “automotive” and more.

- **OS-A Platform**: The OS-A platform specifies a category of rich-OS platforms that support operating systems like Linux, FreeBSD, Windows and more; flavors that run on enterprise and embedded class application processors. Each OS-A platform that is defined below is independent in its representation and is not dependent on any other platform for its features or specifications. Requirements common across multiple platforms are bundled together in the OS-A Common Requirements section in order to prevent duplication of content. The specific platform can include all or some of the requirements in the common section and add or modify these as per the specific requirements. The OS-A platforms that are currently defined are the following:
  - **OS-A Embedded Platform**
  - **OS-A Server Platform**
- **M Platform**: The M platform specifies an RTOS platform for bare-metal applications and small operating systems running on a microcontroller. The M platform has a base feature set and extensions as shown below:
  - **Base**
  - **Physical Memory Protection (PMP) Extension**

The current version of this platform spec targets the standardization of functionality available in S, U, VS and VU modes, and the standardization of the SBI (Supervisory Binary Interface as defined in [6]) between Supervisor level (S-mode/VS-mode) and M-mode/HS-mode respectively.

Chapter 2. OS-A Common Requirements

2.1. ISA Requirements

2.1.1. General

- This OS-A platform must comply with the RVA22U and RVA22S ISA profiles as defined in the RISC-V ISA Profiles specification [11].
- A non-conforming extension that conflicts with a supported standard extensions must satisfy at least one of the following:
  - It must be disabled by default.
  - The supported standard extension must be declared as unsupported in all feature discovery structures used by software. This option is allowed only if the standard extension is not required.
- All hart PMA regions for main memory must be marked as coherent.
- Memory accesses by I/O masters can be coherent or non-coherent with respect to all hart-related caches.

2.1.2. Supervisor mode

- sstatus
  - sstatus.UBE must support the same access attribute (read-only or writable) as mstatus.UBE.
- stvec
  - Both direct and vectored modes must be supported.
  - The alignment constraint for BASE fields must be at most 256B.
- scounteren
  - Writeable bits must be implemented for all supported (not hardwired to zero) hpmcounters.
- stval
  - stval must not be hardwired to 0 and in all cases must be written with non-zero and zero values as architecturally defined.
- satp
  - For RV32, Bare and Sv32 translation modes must be supported.
  - For RV64, Bare and Sv39 translation modes must be supported.

2.1.3. Hypervisor extension

- hstatus
  - VTW bit must not be hardwired to 0.
  - VTVM bit must not be hardwired to 0.
- hcounteren

    - Writeable bits must be implemented for all supported (not hardwired to zero) hpmcounters.
- htval
    - htval must not be hardwired to 0 and in all cases must be written with non-zero and zero values as architecturally defined.
- htinst/mtinst
    - htinst and mtinst must not be hardwired to 0 and must be written with a transformed instruction (versus zero) when defined and allowed architecturally.
- hgatp
    - For RV32, Bare and Sv32x4 translation modes must be supported.
    - For RV64, Bare and Sv39x4 translation modes must be supported.
- vstvec
    - Both direct and vectored modes must be supported.
    - The alignment constraint for BASE fields must be at most 256B.
- vstval
    - vstval must not be hardwired to 0 and in all cases must be written with non-zero and zero values as architecturally defined.
- vsatp
    - For RV32, Bare and Sv32 translation modes must be supported.
    - For RV64, Bare and Sv39 translation modes must be supported.

2.2. Debug

The OS-A platform common requirements are the following:

- Implement resethaltreq
    - Rationale: Debugging immediately out of reset is a useful debug tool. The resethaltreq mechanism provides a standard way to do this.
- Implement the program buffer
    - Rationale: The program buffer is easier for most implementations than abstract access.
    - Rationale: Debuggers need to be able to insert ebreak instructions into memory and make sure that the ebreak is visible to subsequent instruction fetches. Abstract access has no support for `fence.i` (or similar mechanisms).
- abstractcs.relaxedpriv must be 0
    - Rationale: Doing otherwise is a potential security problem.
- abstractauto must be implemented
    - Rationale: autoexecprogbuf allows faster instruction-stuffing.
    - Rationale: autoexecdata allows fast read/write of a region of memory.
- dcsr.mprven must be tied to 1
    - Rationale: Emulating two-stage table walks and PMP checks and endianness swapping is a heavy burden on the debugger.

- In textra, sselect must support the value 0 and either value 1 or 2 (or both)
  - Rationale: There must be some way to limit triggers to only match in a particular user context and a way to ignore user context.
- If textra.sselect=1 is supported, the number of implemented bits of svalue must be at least the number of implemented bits of scontext
  - Rationale: This allows matching on every possible scontext.
- If textra.sselect=2 is supported, the number of implemented bits of svalue must be at least ASIDLEN to match every possible ASID
- In textra, mhselect must support the value 0. If the H extension is supported then mhselect must also support either values 1 and 5 or values 2 and 6 (or all four)
  - Rationale: There must be some way to limit triggers to only match in a particular guest context and a way to ignore guest context.
- If textra.mhselect=1,5 are supported and if H is the number of implemented bits of hcontext then, unless all bits of mhvalue are implemented, at least H-1 bits of mhvalue must be implemented
  - Rationale: This allows matching on every possible hcontext (up to the limit of the field width). It is H-1 bits instead of H because mhselect[2] provides one bit.
- If textra.mhselect=2,6 are supported, the number of implemented bits of mhvalue must be at least VMIDLEN-1
  - Rationale: This allows matching on every possible VMID. It is VMIDLEN-1 instead of VMIDLEN because mhselect[2] provides one bit.
- Implement at least four mcontrol6 triggers that can support matching on PC (select=0, execute=1, match=0) with timing=0 and full support for mode filtering (vs, vu, m, s, u) for all supported modes and support for textra as above
  - Rationale: The debugger needs breakpoints and 4 is a sufficient baseline.
- Implement at least four mcontrol6 triggers that can support matching on load and store addresses (select=0, match=0, and all combinations of load/store) with timing=0 and full support for mode filtering (vs, vu, m, s, u) for all supported modes and support for textra as above
  - Rationale: The debugger needs watchpoints and 4 is a sufficient baseline.
- Implement at least one trigger capable of icount and support for textra as above for self-hosted single step needs this
- Implement at least one trigger capable of etrigger and support for textra as above to catch exceptions
- Implement at least one trigger capable of itrigger and support for textra as above to catch interrupts
- The minimum trigger requirements must be met for action=0 and for action=1 (possibly by the same triggers)
  - Rationale: The intent is to have full support for external debug and full support for self-hosted debug (though not necessarily at the same time). This can be provided via the same set of triggers or separate sets of triggers. External debug support for icount is unnecessary due to dcsr.step and is therefore called out separately.
- For implementations with multiple cores, support for at least one halt group and one resume group (in addition to group 0)

    - Rationale: Allows stopping all harts (approximately) simultaneously which is useful for debugging MP software.
- dcsr.stepie must support the 0 setting. It is optional to support the 1 setting
    - Rationale: It is not generally useful to step into interrupt handlers.
- dcsr.stopcount must be supported and the reset value must be 1
    - Rationale: The architecture has strict requirements on minstret which may be perturbed by an external debugger in a way that's visible to software. The default should allow code that's sensitive to these requirements to be debugged.

2.3. Timers

- One or more ACLINT MTIMER devices are required for the OS-A platform.
- Platform must support an ACLINT MTIME counter resolution of 100ns or less (corresponding to a clock tick frequency of at least 10 MHz).

2.4. Interrupts

The OS-A platform must comply with one of the four interrupt support categories described in following sub-sections. The hardware must support at least one of the four interrupt categories while software must support all of the interrupt categories described below. Any hardware requirement for a specific privilege mode is only applicable for platforms supporting that privilege mode.

2.4.1. Legacy wired IRQs - DEPRECATED

- One or more PLIC devices are required to support wired interrupts.
- One or more ACLINT MSWI devices are required to support M-mode software interrupts.
- Software interrupts for S-mode and VS-mode are supported using the SBI IPI extension.
- This category is compatibile with legacy platforms having PLIC plus CLINT devices.
- MSI external interrupts are not supported.
- MSI virtualization is not supported.

2.4.2. Only Wired IRQs

- One or more AIA APLIC devices are required to support wired interrupts.
- One or more ACLINT MSWI devices are required to support M-mode software interrupts.
- One or more ACLINT SSWI devices are required to support S/HS-mode software interrupts.
- Software interrupts for VS-mode are supported using the SBI IPI extension.
- MSI external interrupts are not supported.
- MSI virtualization is not supported.

2.4.3. MSIs and Wired IRQs

- AIA local interrupt CSRs must be supported by each hart.
  - `siselect` CSR must support holding 9-bit value.
  - `vsiselect` CSR must support holding 9-bit value if H-extension is implemented.
- Per-hart AIA IMSIC devices must support MSIs for M-mode and S/HS-mode.
  - Must support IPRIOLEN = 6 to 8.
  - Must support at least 63 distinct interrupt identities.
  - Must implement `seteipnum_le` memory-mapped register.
- One, or more AIA APLIC devices to support wired interrupts if the platform support wired irqs.
  - EIID and IID fields must be 6 to 8 bits wide matching the number of interrrupt identities supported by AIA IMSIC.
- Software interrupts for M-mode and S/HS-mode are supported using AIA IMSIC devices.
- Software interrupts for VS-mode are supported using the SBI IPI extension.
- MSI virtualization is not supported.

2.4.4. MSIs, Virtual MSIs, and Wired IRQs

- To support virtual MSIs, the H-extension must be implemented.
  - GEILEN must be 3 or more.
- AIA local interrupt CSRs must be supported by each hart.
  - `siselect` CSR must support holding 9-bit value.
  - `vsiselect` CSR must support holding 9-bit value.
- Per-hart AIA IMSIC devices are required to support MSIs for M-mode, HS-mode and VS-mode.
  - Must support IPRIOLEN = 6 to 8.
  - Must support at least 63 distinct interrupt identities.
  - Must implement `seteipnum_le` memory-mapped register.
  - Must implement at least 3 guest interrupt files.
- One, or more AIA APLIC devices are required to support wired interrupts if the platform support wired irqs.
  - EIID and IID fields must be 6 to 8 bits wide matching the number of interrrupt identities supported by AIA IMSIC.
- Software interrupts for M-mode, HS-mode and VS-mode are supported using AIA IMSIC devices.
- MSI virtualization is supported.

2.4.5. Summary

The Table 1 below summarizes the four categories of interrupt support and timer support allowed on an OS-A platorm.

Table 1. Interrupts and Timer support in OS-A platforms

OS-A Platform	MSIs			Wired Interrupts			Software Interrupts			Timer		
	M-mode	S-mode	VS-mode	M-mode	S-mode	VS-mode	M-mode	S-mode	VS-mode	M-mode	S-mode	VS-mode
Legacy Wired IRQs	NA	NA	NA	PLIC	PLIC	PLIC (emulate)	MSWI	SBI IPI	SBI IPI	MTIMER	SBI Timer	SBI Timer
Only Wired IRQs	NA	NA	NA	APLIC	APLIC	APLIC (emulate)	MSWI	SSWI	SBI IPI	MTIMER	Priv Sstc	Priv Sstc
MSIs and Wired IRQs	IMSIC	IMSIC	IMSIC (emulate)	APLIC	APLIC	APLIC (emulate)	IMSIC	IMSIC	SBI IPI	MTIMER	Priv Sstc	Priv Sstc
MSIs, Virtual MSIs and Wired IRQs	IMSIC	IMSIC	IMSIC	APLIC	APLIC	APLIC (emulate)	IMSIC	IMSIC	IMSIC	MTIMER	Priv Sstc	Priv Sstc

2.5. System Peripherals

2.5.1. UART/Serial Console

In order to facilitate the bring-up and debug of the low level initial platform, hardware is required to implement a UART port that confirms to the following requirements and firmware must support the console using this UART:

- The UART register addresses are required to be aligned to 4 byte boundaries. If the implemented register width is less than 4 bytes then the implemented bytes are required to be mapped starting at the smallest address.
- The UART port implementation is required to be register-compatible with one of the following:
  - UART 16550 - MANDATORY
  - UART 8250 - DEPRECATED

2.6. Runtime Services

2.6.1. SBI

- The M-mode runtime must implement SBI specification [6] or higher.
- Required SBI extensions include:
  - SBI TIME
  - SBI IPI
  - SBI RFENCE
  - SBI HSM
  - SBI SRST
  - SBI PMU

2.6.2. UEFI

- Wherever applicable UEFI firmware must implement UEFI interfaces over similar interfaces and services present in the SBI specification. For example, the UEFI ResetSystem() service must be

implemented via the SBI System Reset Extension.

- The operating system should prioritize calling the UEFI interfaces before the SBI or platform specific mechanisms.

2.7. Software and ABIs

The platform specification mandates the following requirements for software components:

- All RISC-V software components must comply with the RISC-V Calling Convention specification [12].
- All RISC-V software components that use ELF files must comply with the RISC-V ELF specification [13].
- All RISC-V software components that use DWARF files must comply with the RISC-V DWARF specification [14].

Rationale: The platform specification intends to avoid fragmentation and promotes interoperability.

- To order older stores before younger instruction fetches, user-level programs must use system-supplied library calls (e.g. GNU libc's `__riscv_flush_icache`, which invokes the Linux kernel's corresponding vDSO routine), rather than executing the `fence.i` instruction directly.

Rationale: The `fence.i` instruction only orders the current hart's instruction fetches - which is insufficient even for single-threaded programs since a thread may migrate to a different hart.

2.8. Security

- If M-mode is supported in the platform, all machine mode assets, such as code and data, shall be protected from all non-machine mode accesses from the harts in the system. Additionally, I/O agent access protection must be required within the system to protect machine mode assets. Therefore, the following requirements are mandatory for platforms with M-mode:
  - Platform must provide a protection mechanism from non-machine mode hart transactions that precisely traps if violated.
  - Platform must provide a protection mechanism from I/O agents manipulating or accessing machine mode assets.

Chapter 3. OS-A Embedded Platform

The OS-A Embedded Platform targets embedded class applications. The OS-A Embedded Platform inherits all the requirements as defined in the OS-A Platform Common Requirements section. Additional requirements are detailed in the following sections.

3.1. PMU

The RVA22 profile defines 32 PMU counters out-of-which first three counters are defined by the privilege specification while other 29 counters are programmable. The SBI PMU extension defines a set of hardware events that can be monitored using these programmable counters. This section defines the minimum number of programmable counters and hardware events required for an OS-A Embedded compatible platform.

- Counters
  - The platform does not require to implement any of the programmable counters.
- Events
  - The platform does not require to implement any of the hardware events defined in SBI PMU extensions.

3.2. Boot Process

- The OS-A Embedded Platform must comply with the EBBR specification [15]. Any deviation from the EBBR will be explicitly mentioned in the requirements in this section.

3.2.1. Firmware

3.2.1.1. Storage and Partitioning

- GPT partitioning required for shared storage.
- MBR support is not required.

3.2.2. Hardware Discovery Mechanisms

- Platforms must support the Unified Discovery specification for all pre-boot information population [20].

3.2.2.1. Device Tree (DT)

- Device Tree (DT) is the required mechanism for the hardware discovery and configuration.

Chapter 4. OS-A Server Platform

The OS-A Server Platform targets server class applications. The OS-A Server Platform inherits all the requirements as defined in the OS-A Platform Common Requirements section. Additional requirements are detailed in the following sections.

4.1. ISA Requirements

4.1.1. General

- The hypervisor H-extension must be supported.
- The Zam extension must be supported for misaligned addresses within at least aligned 16B regions.
- The `time` CSR must be implemented in hardware.
- The Sstc extension [5] must be implemented.

Recommendation
There should be hardware support for all misaligned accesses; misaligned accesses should not take address misaligned exceptions.

4.1.2. Supervisor mode

- satp
  - For RV64, Sv48 translation mode must be supported.
  - At least 8 ASID bits must be supported and not hardwired to 0.

4.1.3. Hypervisor extension

- hgatp
  - For RV64, Sv48x4 translation mode must be supported.
  - At least 8 VMID bits must be supported and not hardwired to 0.
- vsatp
  - For RV64, Sv48 translation mode must be supported.
  - At least 8 ASID bits must be supported and not hardwired to 0.

4.2. PMU

The RVA22 profile defines 32 PMU counters out-of-which first three counters are defined by the privilege specification while other 29 counters are programmable. The SBI PMU extension defines a set of hardware events that can be monitored using these programmable counters. This section defines the minimum number of programmable counters and hardware events required for an OS-A Server compatible platform.

- Counters

    - The platform must implement at least 8 programmable counters.
- Events
    - Hardware general events
        - The platform must implement all of the general hardware events defined by the SBI PMU extension.
    - Hardware cache events
        - The platform must implement READ operations for all of the hardware cache events except SBI_PMU_HW_CACHE_NODE and SBI_PMU_HW_CACHE_LL defined in the SBI PMU extension.
        - The platform must implement WRITE operation for L1D, and DTLB caches.

Implementation Note

Any platform that does not implement the micro-architectural features related to a hardware event may hardwire the event value to zero.

4.3. Debug

The OS-A Server platform includes all the requirements as specified in the OS-A Common Requirements section plus the following:

- Implement at least six mcontrol6 triggers that can support matching on PC (select=0, execute=1, match=0) with timing=0 and full support for mode filtering (vs, vu, m, s, u) for all supported modes and support for textra as above
    - Rationale: Other architectures have found that 4 breakpoints are insufficient in more capable systems and recommend 6.
- If system bus access is implemented then accesses must be coherent with respect to all harts connected to the DM
    - Rationale: Debuggers must be able to view memory coherently.

4.4. Interrupts

The OS-A Server platform must support the interrupt requirements as specified in the OS-A Common Requirements Interrupts section Section 2.4.4 plus the following:

- The H-extension implemented by each hart must support GEILEN = 5 or more.
- Per-hart AIA IMSIC devices.
    - Must support at least 255 distinct interrupt identities.
    - Must support IPRIOLEN = 8.
- EIID and IID fields of AIA APLIC devices must be at least 8 bits wide matching the number of interrupt identities supported by AIA IMSIC.

Recommendation
Platforms should implement at least 5 guest interrupt files. More guest interrupt files allow for better

VM oversubscription on the same hart.

4.5. Boot Process

4.5.1. Firmware

The boot and system firmware for the server platforms must support UEFI as defined in the section 2.6.1 of the UEFI Specification [1] with some additional requirements described in following sub-sections.

4.5.1.1. UEFI Configuration Tables

The platforms are required to provide following tables:

- **EFI_ACPI_20_TABLE_GUID** ACPI configuration table which is at version 6.4+ or newer with HW-Reduced ACPI model.
- **SMBIOS3_TABLE_GUID** SMBIOS table which conforms to version 3.4 or later.

4.5.1.2. UEFI Protocol Support

The UEFI protocols listed below are required to be implemented.

Table 2. Additional UEFI Protocols

Protocol	UEFI Section	Note
EFI_PCI_ROOT_BRIDGE_IO_PROTOCOL	14	For PCIe support
EFI_PCI_IO_PROTOCOL	14.4	For PCIe support

4.5.2. Hardware Discovery Mechanisms

- Platforms must support the Unified Discovery specification for all pre-boot information population [20].

4.5.2.1. ACPI

ACPI is the required mechanism for the hardware discovery and configuration. Server platforms are required to adhere to the RISC-V ACPI Platform Requirements Specification [21]. Platform firmware must support ACPI and the runtime OS environment must use ACPI for device discovery and configuration.

4.5.2.2. SMBIOS

The System Management BIOS (SMBIOS) table is required for the platform conforming to server extension. The SMBIOS records provide basic hardware and firmware configuration information used widely by the platform management applications.

The SMBIOS table is identified using **SMBIOS3_TABLE_GUID** in UEFI configuration table. The memory type used for the SMBIOS table is required to be of type **EfiRuntimeServicesData**.

In addition to the conformance guidelines as mentioned in **ANNEX A / 6.2** of the SMBIOS

specification 3.4.0, below additional structures are required.

Table 3. Required SMBIOS structures

Structure Type	SMBIOS Section	Note
Management Controller Host Interface (Type 42)	7.43	Required for Redfish Host Interface.
Processor Additional Information (Type 44)	7.45	This structure provides the additional information of RISC-V processor characteristics and HART hardware features discovered during the firmware boot process.

4.6. Runtime services

The OS-A Server platform includes all the runtime services requirements as specified in the OS-A Common Requirements Runtime Services section plus the following.

4.6.1. UEFI

The UEFI run time services listed below are required to be implemented.

Table 4. Required UEFI Runtime Services

Service	UEFI Section	Note
GetVariable	8.2	
GetNextVariableName	8.2	
SetVariable	8.2	A dedicated storage for firmware is required so that there is no conflict in access by both firmware and the OS.
QueryVariableInfo	8.2	
GetTime	8.3	System Date/Time accessed by the OS and firmware.(Refer to System Date/Time section)
SetTime	8.3	System Date/Time set by the OS and firmware.(Refer to System Date/Time section)
GetWakeupTime	8.3	Interface is required to be implemented but it can return EFI_UNSUPPORTED.(Refer to System Date/Time section)

Service	UEFI Section	Note
SetWakeupTime	8.3	Interface is required to be implemented but it can return EFI_UNSUPPORTED.(Refer to System Date/Time section)
SetVirtualAddressMap	8.4	
ConvertPointer	8.4	
GetNextHighMonotonicCount	8.5	
ResetSystem	8.5	If SBI SRST implementation is also available, the OS should not use the SBI interface directly but use this UEFI interface.
UpdateCapsule	8.5	Interface is required to be implemented but it can return EFI_UNSUPPORTED.
QueryCapsuleCapabilities	8.5	Interface is required to be implemented but it can return EFI_UNSUPPORTED.

4.7. System Peripherals

The OS-A Server platform includes all the system peripheral requirements as specified in the OS-A Common Requirements System Peripherals section plus the added requirements in this section.

4.7.1. Watchdog Timers

Implementation of a two-stage watchdog timer, as defined in the RISC-V Watchdog Timer Specification[22] is required. Software must periodically refresh the watchdog timer, otherwise a first-stage watchdog timeout occurs. If the watchdog timer remains un-refreshed for a second period, then a second-stage watchdog timeout occurs.

If a first-stage watchdog timeout occurs, a Supervisor-level interrupt request is generated and sent to the system interrupt controller, targeting a specific hart.

If a second-stage watchdog timeout occurs, a system-level interrupt request is generated and sent to a system component more privileged than Supervisor-mode such as:

- The system interrupt controller, with a Machine-level interrupt request targeting a specific hart.
- A platform management processor.
- Dedicated reset control logic.

The resultant action taken is platform-specific.

4.7.2. System Date/Time

In order to facilitate server manageability, server extension platform is required to provide the mechanism to maintain system date/time for UEFI runtime Time service.

- UEFI Runtime Time Service
  - GetTime()
    Must be implemented by firmware to incorporate with the underlying system date/time mechanism.
  - SetTime(), GetWakeupTime() and SetWakeupTime()
    These Time services must be implemented but allowed to return EFI_UNSUPPORTED if the platform doesn't require the features or the system date/time mechanism doesn't have the capabilities.

4.7.3. PCIe

Platforms are required to support at least PCIe Base Specification Revision 1.1 [24].

4.7.3.1. PCIe Config Space

- Platforms must support access to the PCIe config space via ECAM as described in the PCIe Base specification.
- The entire config space for a single PCIe domain must be accessible via a single ECAM I/O region.
- Platform firmware must implement the MCFG table as listed in the ACPI System Description Tables above to allow the operating systems to discover the supported PCIe domains and map the ECAM I/O region for each domain.
- Platform software must configure ECAM I/O regions such that the effective memory attributes are that of a PMA I/O region (i.e. strongly-ordered, non-cacheable, non-idempotent).

4.7.3.2. PCIe Memory Space

Platforms are required to map PCIe address space directly in the system address space. The physical addresses used by the hart for outbound accesses must not undergo any further translation/offsetting and must be sent to the PCIe device unmodified.

The unmodified physical address in an inbound accesses may optionally be presented to an IOMMU for address tranlation. If no IOMMU is employed for address translation then the unmodified physical address sent by the device must be used for accessing system memory. If an IOMMU is employed then the unmodified translated address provided by the IOMMU must be used for accessing system memory.

- PCIe Outbound Memory
  PCIe devices and bridges/switches frequently implement BARs which only support 32-bit addressing or support 64 bit addressing but do not support prefetchable memory. To support mapping of such BARs, platforms are required to reserve some space below 4G for each root port present in the system.

> ***Implementation Note***
> *Platform software would likely configure these per root port regions such that their effective memory attributes are that of a PMA I/O region (i.e. strongly-ordered, non-cacheable, non-idempotent). Platforms would likely also reserve some space above 4G to map BARs that support 64 bit addressing and prefetchable memory which could be configured by the platform software as either I/O or memory.*

- PCIe Inbound Memory
  For security reasons, platforms with M-mode must provide a mechanism controlled by M-mode software to restrict inbound PCIe accesses from accessing regions of address space intended to be accessible only to M-mode software.

> ***Implementation Note***
> *Such an access control mechanism could be analogous to the per-hart PMP as described in the RISC-V Privileged Architectures specification.*

4.7.3.3. PCIe Interrupts

- Platforms must support Message Signaled (MSI or MSI-X) Interrupts.
- Platforms may optionally support INTx interrupt signaling.
- Following are the requirements for INTx interrupt signaling if supported:
  - For each root port in the system, the platform must map all the INTx virtual wires to four distinct sources at the APLIC. Each of these sources must be configured as Level0 as described in Table 4.2 (Encoding of the SM (Source Mode) field) of the RISC-V AIA specification.
  - Platform firmware must implement the _PRT as described in section 6.2.13 of ACPI Specification to describe the mapping of interrupt pins and the corresponding interrupt minor identities at the Hart.
  - If interrupt generation for correctable/fatal/non-fatal error messages is enabled via the root error command register of the AER capability and the root port does not support MSI/MSI-X capability, then the platform is required to generate an INTx interrupt via the APLIC.
- Following are the requirements for MSI:
  - As per the RISC-V AIA specification, since the number 0 is not a valid interrupt identity, the platform software is required to ensure that MSI data value assigned to a PCIe function is never 0. For e.g for a PCIe function which requests 16 MSI vectors the minimum MSI data value assigned by the platform software can be 0x10 so that the function can use lower 4 bits to assert each of the 16 vectors.

4.7.3.4. PCIe cache coherency

Memory that is cacheable by harts may not be kept coherent by hardware when PCIe transactions to that memory are marked with a No_Snoop bit of one. On platforms that honour No_Snoop bit, software must manage coherency on such memory; otherwise, software coherency management is not required.

4.7.3.5. PCIe Topology

Platforms are required to implement at least one of the following topologies and the components required in that topology.

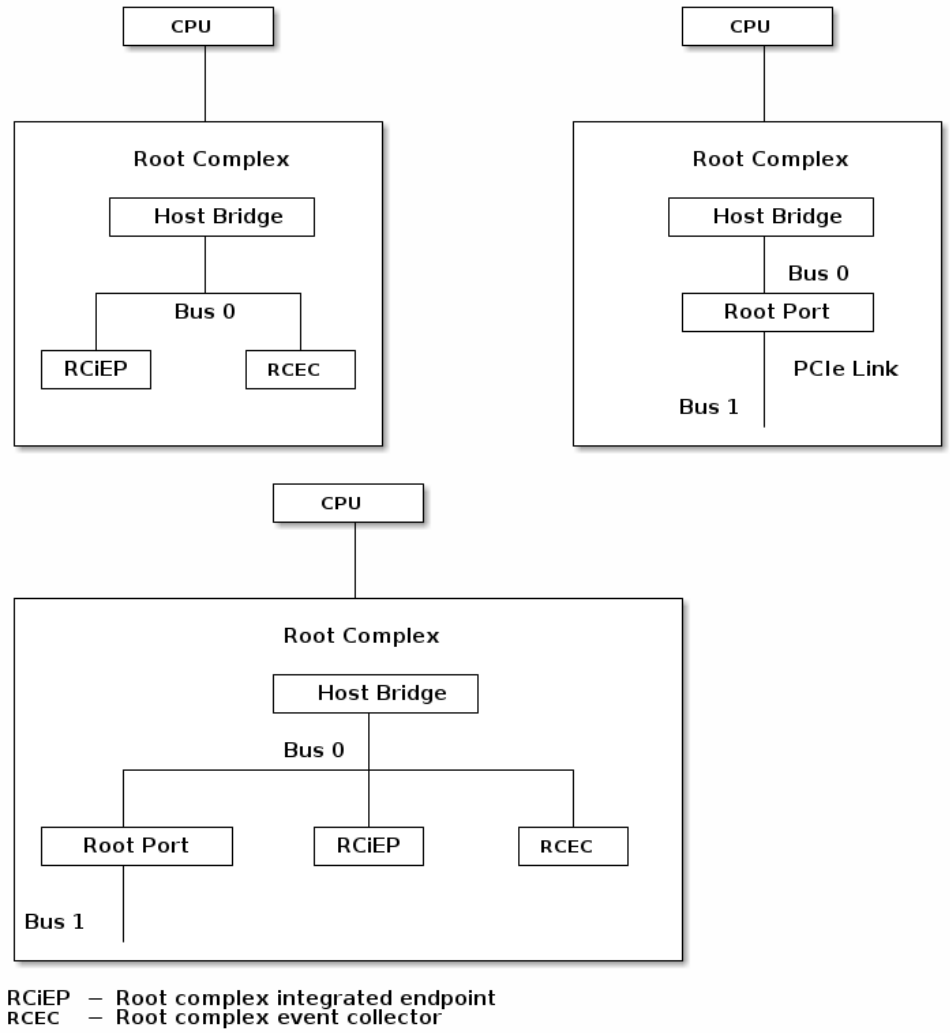

Figure 1. PCIe Topologies

- Host Bridge
  Following are the requirements for host bridges:
  - Any read or write access by a hart to an ECAM I/O region must be converted by the host bridge into the corresponding PCIe config read or config write request.
  - Any read or write access by a hart to a PCIe outbound region must be forwarded by the host bridge to a BAR or prefetch/non-prefetch memory window, if the address falls within the region claimed by the BAR or prefetch/ non-prefetch memory window. Otherwise the host bridge must return an error.
  - Host bridge must return all 1s in the following cases:
    - Config read to non existent functions and devices on root bus.
    - Config reads that receive Unsupported Request response from functions and devices on the root bus.
- Root ports
  Following are the requirements for root ports.
  - Root ports must appear as PCI-PCI bridge to software.
  - Root ports must implement all registers of Type 1 header.
  - Root ports must implement all capabilities specified in the PCIe Base specification for a root

port.

- Root ports must forward type 1 configuration access when the bus number in the TLP is greater than the root port's secondary bus number and less than or equal to the root port's subordinate bus number.
- Root ports must convert type 1 configuration access to a type 0 configuration access when bus number in the TLP is equal to the root port's secondary bus number.
- Root ports must respond to any type 0 configuration accesses it receives.
- Root ports must forward memory accesses targeting its prefetch/non-prefetch memory windows to downstream components. If address of the transaction does not fall within the regions claimed by prefetch/non-prefetch memory windows then the root port must generate a Unsupported Request.
- Root port requester id or completer id must be formed using the bdf of the root port.
- The root ports must support the CRS software visibility.
- The root port must implement the AER capability.
- Root ports must return all 1s in the following cases:
  - Config read to non existent functions and devices on secondary bus.
  - Config reads that receive Unsupported Request from downstream components.
  - Config read when root port's link is down.

- RCiEP
  All the requirements for RCiEP in the PCIe Base specification must be implemented. In addition the following requirements must be met:
  - If RCiEP is implemented then RCEC must be implemented as well. All requirements for RCEC specified in the PCIe Base specification must be implemented. RCEC is required to terminate the AER and PME messages from RCiEP.

4.7.3.6. PCIe Device Firmware

PCI expansion ROM code type 3 (UEFI) image must be provided by PCIe device platform according to PCI Firmware Specification [19] if that PCIe device is utilized during UEFI firmware boot process. The image stored in PCI expansion ROM is a UEFI driver that must be compliant with UEFI specification [1] 14.4.2 PCI Option ROMs.

4.8. Security

The OS-A Server platform includes all the security requirements as specified in the OS-A Common Requirements security section plus the following:

- Support for some form of Secure Boot, as a means to ensure the integrity of platform firmware and software, is required. Flexibility is provided as to the many details and implementation approaches. Future platform specs are expected to standardize some or many of these aspects. For now, it is recommended that the following security properties are met:
  - The secure boot process is rooted in dedicated hardware.
  - Cryptographic algorithms are independently validated or certified for implementation correctness.

- The combination of key length and cryptographic algorithm provides suitable security strength.
- A cryptographically secure entropy source (or multiple entropy sources) is used in key material generation and monitoring of entropy source's health is implemented.
- Critical security parameters are securely stored and only accessible with appropriate privileges.
- Authorization is required for any modifications to the platform secure boot configuration.
- It is clearly understood what aspects of the platform boot process are protected by secure boot.
- If M-mode is supported in the platform, all machine mode assets, such as code and data, shall be protected from all non-machine mode accesses from the harts in the system. Additionally, I/O agent access protection must be required within the system to protect machine mode assets. Therefore, the following requirements are mandatory for platforms with M-mode:
  - Platform must provide a protection mechanism from non-machine mode hart transactions that precisely traps if violated.
  - Platform must provide a protection mechanism from I/O agents manipulating or accessing machine mode assets.

4.9. RAS

All the below mentioned RAS features are required for the OS-A platform server extension:

- Main memory must be protected with SECDED-ECC or a stronger/advanced method of protection.
- Cache structures must be protected. The protection mechanisms may include single-bit/multi-bit error detection/correction schemes.
- There must be memory-mapped RAS registers associated with these protected structures to log detected errors with information about the type and location of the error.
- The platform must support the APEI specification to convey all error information to OSPM.
- Correctable errors must be reported by hardware and either be corrected or recovered by hardware, transparent to system operation and to software.
- Hardware must provide status of these correctable errors via RAS registers.
- Uncorrectable errors must be reported by the hardware via RAS error registers for system software to take the needed corrective action.
- Attempted use of corrupted uncorrectable data must result in an exception with a distinguishing custom exception code; preferably a precise exception on that instruction if possible.
- The platform should provide the capability to configure RAS errors to trigger firmware-first or OS-first error interrupt.
- Errors logged in RAS registers must be able to generate an interrupt request to the system interrupt controller that may be directed to either M-mode or S/HS-mode for firmware-first or OS-first error reporting.
- If the RAS error is handled by firmware, the firmware should be able to choose to expose the error to S/HS mode for further processing or just hide the error from S/HS software.
- If the RAS event is configured as the firmware first model, the platform should be able to trigger the highest priority of M-mode interrupt to all HARTs in the physical RV processor.
- Logging and/or reporting of errors can be masked.

- PCIe AER capability is required.

Chapter 5. M Platform

5.1. Scope

The M Platform specification aims to apply to a range of embedded platforms. In this case embedded platforms range from hand coded bare metal assembly all the way to to embedded operating systems such as Zephyr and embedded Linux.

This specification has two competing interests. On one hand embedded software will be easier to write and port if all the embedded hardware is similar. On the other hand vendors want to differentiate their product and reuse existing IP and SoC designs.

Due to this, the M Platform specification has both required and recommended components. All required components must be met in order to meet this specification. It's strongly encouraged that all recommended components are met as well, although they do not have to in order to meet the specification.

5.2. Base

5.2.1. Architecture

The M Platform must comply with the RVM22M profile defined by the RISC-V profiles specification [11].

5.2.2. Interrupt Controller

Embedded systems are recommended to use a spec compliant PLIC [7], a spec compliant CLIC [8] or both a CLIC and and PLIC.

If using just a PLIC the system must continue to use the original basic `xsip`/`xtip`/`xeip` signals in the `xip` register to indicate pending interrupts. If using the CLIC then both the original basic and CLIC modes of interrupts must be supported.

Embedded systems cannot use a non-compliant interrupt controller and still call it a PLIC or CLIC.

5.2.3. Timer Support

The M Platform must implement one or more RISC-V ACLINT MTIMER [9] devices. This will provide the `mtime` and `mtimecmp` memory mapped registers as required by the RISC-V privilege specification [4].

The `mcounteren`.TM and `scounteren`.TM bits *must not* be hardwired, regardless as to whether accesses to the `time` CSR are implemented directly or via traps.

5.2.4. Memory Map

It is recommended that main memory and loadable code (not ROM) start at address `0x8000_0000`.

5.3. Physical Memory Protection (PMP) Extension

It is recommended that any system that implement more than just machine mode also implement PMP support.

When PMP is supported it is recommended to include at least 4 regions, although if possible more should be supported to allow more flexibility. Hardware implementations should aim for supporting at least 16 PMP regions.

References

- [1] UEFI Specification, Version: v2.9
- [2] Devicetree Specification, Version: v0.3
- [3] RISC-V Unprivileged Architecture Specification, Version:20191214-draft
- [4] RISC-V Privileged Architecture Specification, Version: v1.12-draft
- [5] RISC-V Privleged Architecture Sstc Extension, Version: Draft
- [6] RISC-V SBI Specification, Version: v0.3
- [7] RISC-V PLIC Specification, Version: v1.0-draft
- [8] RISC-V CLIC Specification, Version: draft-bc89a5e3d61d
- [9] RISC-V ACLINT Specification, Version: v1.0-draft2
- [10] RISC-V AIA Specification, Version: v0.2-draft.24
- [11] RISC-V Profiles Specification, Version: draft-8e8951987e2a
- [12] RISC-V Calling Convention specification, Version: 1.0-rc1
- [13] RISC-V ELF specification, Version: 1.0-rc1
- [14] RISC-V DWARF specification, Version: 1.0-rc1
- [15] EBBR Specification, Version: v2.0.1
- [16] ACPI Specification, Version: v6.4
- [17] APEI Specification, Version: v6.4
- [18] SMBIOS Specification, Version: v3.4.0
- [19] PCI Firmware Specification, Version: 3.3
- [20] Unified Discovery Specification (TBD)
- [21] RISC-V ACPI Platform Requirements Specification, Version: Draft-20210812
- [22] RISC-V Watchdog Timer Specification, Version: Version 1.0
- [23] RISC-V Platform Platform Policy, Version: 1.0
- [24] PCIe Base Specification Revision, Revision: 1.1