Arm® PC Base System Architecture 1.0

Platform Design Document

Non-confidential

Contents

Release information

Version 1 (20 Oct 2024)

- Initial public release.

Arm Non-Confidential Document License ("License")

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Arm document reference: PRE-21585

version 5.0, March 2024

About this document

Terms and abbreviations

Term	Meaning
Arm ARM	Arm Architecture Reference Manual. See [1].
Base PC System	A system that is compliant with the PC Base System Architecture.
BBR	Base Boot Requirements. See [2].
BSA	Base System Architecture. See [3].
Completer	An agent in a computing system that responds to and completes a memory transaction that was initiated by a Requester.
DMA	Direct Memory Access.
DRTM	Dynamic Root of Trust for Measurement.
ECAM	Enhanced Configuration Access Mechanism.
GIC	Generic Interrupt Controller.
LPI	Locality-specific Peripheral Interrupt (GICv3 [4]).
OTP	One-Time Programmable memory.
PC	Personal Computer.
PC-BSA	PC Base System Architecture.
PE	Processing Element, as defined in the Arm ARM.
PMU	Performance Monitor Unit.
PPI	Private Peripheral Interrupt.
Requester	An agent in a computing system that is capable of initiating memory transactions.
SGI	Software-Generated Interrupt.
SPI	Shared Peripheral Interrupt.
SVM	Shared Virtual Memory.
System firmware data	System description data structures, for example ACPI or Flattened Device Tree.
TCB	Trusted computing base.
TCG	Trusted Computing Group.
TPM	Trusted Platform Module is a technology, typically implemented through a Secure microcontroller, that can securely store artifacts used to authenticate a platform, or platform components. The TPM technical specification is maintained by the TCG consortium.
VM	Virtual Machine.

References

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *DDI 0487 Arm® Architecture Reference Manual for A-profile architecture*. Arm Ltd.

[2] *DEN 0044 Arm® Base Boot Requirements*. Arm Ltd.

[3] *DEN 0094 Arm® Base System Architecture*. Arm Ltd.

[4] *IHI 0069 Arm® Architecture Specification, GIC architecture version 3.0 and version 4.0*. Arm Ltd.

[5] *PCI Express Base Specification Revision 6.0, version 1.0*. PCI-SIG.

[6] *Enhanced Allocation (EA) for Memory and I/O Resources*. PCI-SIG.

[7] *NIST Special Publication 800-57 Part 1 Revision 5*. NIST.

[8] *TCG PC Client Platform TPM Profile Specification for TPM 2.0 version 1.05 or later*. TCG.

Rules-based writing

This specification consists of a set of individual *content items*. A content item is classified as one of the following:

- Declaration
- Rule
- Goal
- Information
- Rationale
- Implementation note
- Software usage

Declarations and Rules are normative statements. An implementation that is compliant with this specification must conform to all Declarations and Rules in this specification that apply to that implementation.

Declarations and Rules must not be read in isolation. Where a particular feature is specified by multiple Declarations and Rules, these are generally grouped into sections and subsections that provide context. Where appropriate, these sections begin with a short introduction.

Arm strongly recommends that implementers read *all* chapters and sections of this document to ensure that an implementation is compliant.

Content items other than Declarations and Rules are informative statements. These are provided as an aid to understanding this specification.

Content item identifiers

A content item may have an associated identifier which is unique among content items in this specification.

After this specification reaches beta status, a given content item has the same identifier across subsequent versions of the specification.

Content item rendering

In this document, a content item is rendered with a token of the following format in the left margin: L_{iiiii}

- *L* is a label that indicates the content class of the content item.
- *iiiii* is the identifier of the content item.

Content item classes

Declaration

A Declaration is a statement that does one or more of the following:

- Introduces a concept
- Introduces a term
- Describes the structure of data
- Describes the encoding of data

A Declaration does not describe behavior.

A Declaration is rendered with the label *D*.

Rule

A Rule is a statement that describes the behavior of a compliant implementation.

A Rule explains what happens in a particular situation.

A Rule does not define concepts or terminology.

A Rule is rendered with the label *R*.

Goal

A Goal is a statement about the purpose of a set of rules.

A Goal explains why a particular feature has been included in the specification.

A Goal is comparable to a “business requirement” or an “emergent property.”

A Goal is intended to be upheld by the logical conjunction of a set of rules.

A Goal is rendered with the label *G*.

Information

An Information statement provides information and guidance as an aid to understanding the specification.

An Information statement is rendered with the label *I*.

Rationale

A Rationale statement explains why the specification was specified in the way it was.

A Rationale statement is rendered with the label *X*.

Implementation note

An Implementation note provides guidance on implementation of the specification.

An Implementation note is rendered with the label *U*.

Software usage

A Software usage statement provides guidance on how software can make use of the features defined by the specification.

A Software usage statement is rendered with the label *S*.

Progressive terminology commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive terms. If you find offensive terms in this document, please contact terms@arm.com.

Feedback

Arm welcomes feedback on its documentation.

If you have any comments or suggestions for additions and improvements create a ticket at

https://support.developer.arm.com.

As part of the ticket include:

- The title (PC Base System Architecture).
- The document ID and version (DEN0151 1.0).
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

1 Personal Computing Base System Architecture

1.1 Arm Base System Architecture

This document is a supplement to the Arm Base System Architecture [3].

1.2 Background

The PC Base System Architecture (PC-BSA) specifies a standard hardware system architecture for Personal Computers (PCs) that are based on the Arm 64-bit Architecture. PC system software, for example operating systems, hypervisors, and firmware can rely on this standard system architecture. PC-BSA extends the requirements specified in the Arm BSA [3].

The PC Base System Architecture embeds the notion of levels of functionality. Each level adds functionality better than the a previous level, adding incremental features that software can rely on. Unless explicitly stated, all specification items that belong to level N apply to levels greater than N.

The list of rules to implement for each PC-BSA level is available in Section 2. All rules of a level, that are described in that level's checklist, are required to be implemented to be compliant to that level.

An implementation is consistent with a level of the PC Base System Architecture if it implements all of the functionality of that level at performance levels that are appropriate for the target uses of that level. This means that all functionality of a level is expected to perform well when exploited by software.

The SBBR recipe in the *Arm Base Boot Requirements (Arm BBR) specification* [2] describes firmware requirements for an Arm PC system. Where this specification refers to system firmware data, it refers to SBBR compliant firmware as specified in the Arm BBR.

1.2.1 PC-BSA levels and minimum component versions

Table 2 summarizes the minimum architecture versions that map to the rules that are mentioned in a PC-BSA level.

This table is indicative only. The rules in each level describe the specific features that are required to be compliant to that level. For a checklist of each level's minimum rules, see Section 2.

Table 2: PC-BSA level mapping summary

Level	PE: A profile	SMMU	GIC
1	v8.1	v3	v3.0

Note

Table 2 shows the version of the architecture extension at which the required features were introduced.

1.3 Level 1

Section 2.1 lists the rules to be implemented for Level 1.

R_P_L1_01 The base PC system must comply with BSA Level 1 requirements as specified by "BSA Level 1 checklist" section from Arm BSA [3].

I For a base PC system that supports secure firmware, it is recommended to follow the additional set of recommendations specified by “Support for Secure Firmware” section from Arm BSA [3]. These recommendations may become required in future versions of this specification.

1.3.1 PE architecture

$R_{P\_L1PE\_01}$ PEs must support 4KB translation granules at stage 1 and stage 2.

I It is recommended that PEs support 64KB translation granules at stage 1 and stage 2.

X Support for 64KB translation granules helps ease the migration blockers between market segments, such as between PCs and servers.

$R_{P\_L1PE\_02}$ PEs must implement 16-bit ASID support.

$R_{P\_L1PE\_03}$ PEs must implement AArch64 at all implemented Exception levels.

$R_{P\_L1PE\_04}$ PEs must comply with FEAT_LSE requirements as specified by B_PE_25 from Arm BSA [3].

I FEAT_LPA [1], Large Physical Address (PA) and Intermediate Physical Address (IPA) support, expands the maximum physical address width from 48 to 52 bits. Using 52-bit PA requires 64KB translation granules.

$R_{P\_L1PE\_05}$ Base PC systems that make use of FEAT_LPA must support a functional system memory map which is wholly contained within the address range from 0 to 2ˆ48-1 (i.e. the first 256TB of physical address space).

U This may be achieved, for example, using a selectable firmware configuration option, or by hiding any mapped resources that exist beyond 256TB address range.

X This is required to support operating systems that do not use the 64KB granule.

I FEAT_LPA2, introduced in Arm v8.7, enables systems to utilize FEAT_LPA for 4KB and 16KB granules. It is recommended that a system that supports FEAT_LPA (for 64KB granules), and supports 4K or 16K granules, should support FEAT_LPA2 as this will prevent the system memory map adjustment described in P_L1PE_05.

I It is consistent with this specification to implement PEs with support for the AArch32 Execution state.

$R_{P\_L1PE\_06}$ If the system contains persistent memory that is exposed to the OS, all PEs must support the clean to point of persistence instruction (DC CVAP). The instruction must be able to perform a clean to the point of persistence for all memory that is exposed as persistent memory to the OS.

$R_{P\_L1PE\_07}$ All PEs must implement FEAT_VMID16.

$R_{P\_L1PE\_08}$ All PEs must implement FEAT_VHE.

1.3.2 Memory map

$R_{P\_L1MM\_01}$ If a base PC system supports 64KB translation granules at stage 2 then it must ensure that:

1. All memory and peripherals can be mapped using 64KB stage 2 pages and must not require the use of 4KB pages at stage 2.
2. All peripherals that can be assigned to different virtual machines will be situated within different 64KB regions of memory.

This rule applies to types of peripherals such as PCIe devices. Here is an indicative list of the peripherals that are exempt from this rule:

- On-chip peripherals whose resources are not managed by PE software, for example system controllers or power management controllers.
- Secure-world peripherals.

X Rule P_L1MM_01 is needed to enable non-secure EL2 hypervisors to use a 64KB translation granule at stage 2 MMU translation.

I It is possible for multiple peripherals that are not assigned to virtual machines to reside within a single 64KB page, as long any peripheral memory does not cross a 64KB page boundary.

1.3.3 Interrupt controller

$R_{P\_L1GI\_01}$ A base PC system must implement at least one interrupt controller that is compliant with the GICv3 or higher architecture [4].

$R_{P\_L1GI\_02}$ All MSI and MSI-X targeting hypervisor and operating system software must be mapped to LPI.

I Rules P_L1GI_01 and P_L1GI_02 provide further restrictions on the allowed interrupt controllers as specified in rules B_GIC_01 and B_GIC_02 in the Arm BSA [3].

$R_{P\_L1GI\_03}$ A GIC that implements Locality-specific Peripheral Interrupts (LPIs) should support clearing GICR_CTLR.EnableLPIs.

$R_{P\_L1GI\_04}$ A GIC implementation that does not support clearing GICR_CTLR.EnableLPIs after it is set must not permit modification of GICR_PENDBASER when GICR_CTLR.EnableLPIs == 1.

1.3.4 PPI assignments

$R_{P\_L1PP\_01}$ The Interrupt IDs must be the same as the recommended values specified by B_PPI_01, B_PPI_02, and B_PPI_03 from Arm BSA [3].

1.3.5 System MMU and device assignment

I Armv8.4 introduces TLB Invalidation instructions which apply to a range of input addresses, instead of just to a single address.

$R_{P\_L1SM\_01}$ If PEs that are used by the base PC system support TLB range instructions, then all OS visible requesters that contain a TLB must support range invalidates. See FEAT_TLBIRANGE in [1].

$R_{P\_L1SM\_02}$ A base PC system must support Stage 1 System MMU functionality. This must be provided by a System MMU that is compliant with the Arm SMMUv3, or higher, architecture revision.

$R_{P\_L1SM\_03}$ A base PC system must support Stage 2 System MMU functionality. This must be provided by a System MMU that is compliant with the Arm SMMUv3, or higher, architecture revision.

X Stage 1 and Stage 2 System MMU functionalities are required for supporting virtualization features such as VM Device Assignment and Shared Virtual Memory (SVM).

$R_{P\_L1SM\_04}$ The integration of the System MMUs must be compliant with rules SMMU_01 and SMMU_02, as specified in "SMMUv3 integration" section from Arm BSA [3].

I The Trusted Computing Base (TCB) of a system consists of all hardware, firmware, and software components that are relied on to enforce the security of the system. A compromise in any TCB component affects the security posture of the system.

$R_{P\_L1SM\_05}$ All DMA capable requesters in a system must pass through SMMU translation, including both Secure and Non-secure devices. The requirement does not apply to DMA capable requesters which are part of the TCB of the system.

I Examples of DMA requesters that are part of the TCB include: SMMUs, interrupt controllers, debug access ports, system controllers, internal root-of-trust processors, management controllers.

X Having SMMUs in front of DMA requesters that are not part of the TCB enables configuration of the system to prevent DMA attacks against boot, DRTM, and the operating system.

1.3.6 Watchdog

I It is recommended that the base PC system implement a Non-secure Generic watchdog as specified by B_WD_01, B_WD_02, B_WD_03, B_WD_04, and B_WD_05 in Arm BSA [3].

1.3.7 Peripheral subsystems

$\mathrm{R}_{\mathrm{P\_L1PCI\_1}}$ All devices that are intended for assignment to a virtual machine are required to be PCIe compliant.

X PCIe [5] provides mechanisms for safe assignment of devices to virtual machines.

$\mathrm{R}_{\mathrm{P\_L1PCI\_2}}$ There must be no OS observable use of PCIe Enhanced Allocation [6].

$\mathrm{R}_{\mathrm{P\_L1NV\_01}}$ Non-volatile storage must be available that can be accessed and updated directly by firmware without the aid of the operating system.

1.3.8 Platform security

$\mathrm{R}_{\mathrm{P\_L1SE\_01}}$ The non-volatile storage for firmware code and protected data (such as UEFI authenticated variables) must be protected from direct modification from a PE running at Non-secure privilege levels. Firmware images and protected data must only be modifiable through authenticated interfaces.

I A platform should have support to detect the rollback of system firmware to an authentic but out of date version.

$\mathrm{R}_{\mathrm{P\_L1SE\_02}}$ Verified boot must be rooted in an on-chip immutable root-of-trust for verification that authenticates the first mutable code using digital signatures. Authentication means:

- Verifying the integrity of the image through a digital signature check
- Verifying that the image signing public key is rooted to a trusted authority

$\mathrm{R}_{\mathrm{P\_L1SE\_03}}$ Hashes and asymmetric keys must have a minimum 128-bit security strength (NIST SP 800-57 [7]).

I Over the course of its life, a system transitions through different stages in the supply chain, such as manufacturing, provisioning, deployed/operational/production, debug, and decommissioned/end-of-life. To allow actors in the supply chain to determine the current security state of a system, the security-relevant state can be reflected in hardware through mechanisms such as fuses and One-Time Programmable (OTP) memory.

$\mathrm{R}_{\mathrm{P\_L1SE\_04}}$ A base PC system must have a hardware-backed IMPLEMENTATION DEFINED mechanism that is the equivalent of OTP memory to reflect the security-relevant state of the system.

$\mathrm{R}_{\mathrm{P\_L1SE\_05}}$ The state described in P_L1SE_04 must be readable by firmware to determine whether a system is in a secure operational/deployed/production state.

I In a deployed/operational/production system lifecycle state, the system must provide a means for secure firmware to detect whether external debug or hardware trace is enabled or not.

1.3.8.1 Trusted Platform Module (TPM)

I A TPM may be implemented as a discrete chip, in a secure enclave, or as a firmware TPM (fTPM) in TrustZone.

$\mathrm{R}_{\mathrm{P\_L1TP\_01}}$ A system must have a TPM implementation that is compliant with the TCG PC Client Platform TPM Profile Specification for TPM 2.0 [8].

$\mathrm{R}_{\mathrm{P\_L1TP\_02}}$ It must not be possible to reset a TPM or SoC independently of each other.

$\mathrm{R}_{\mathrm{P\_L1TP\_03}}$ The platform interface to access the TPM must support localities 0 - 4 of the TPM.

$\mathrm{R}_{\mathrm{P\_L1TP\_04}}$ The system must support hardware-based enforcement to protect locality 4 of the TPM, and it must not be possible for locality 4 to be accessed by Non-secure privilege levels.

I The Arm architecture [1] defines a security model that allow system developers to partition device hardware and software resources, so that they exist in either the Secure world for sensitive resources, or the Normal world for everything else. The Secure world isolation defined by the Arm architecture meets the hardware-based enforcement requirement of P_L1TP_04.

1.3.9 System Management Services

I This specification uses the term System Management Services to refer to a set of ancillary functions which might need to be active while the PE is not in the RUN state. This includes, for example, battery charging, fan and thermal control, keyboard controller, and physical buttons. In some PC systems, such functionality is typically hosted on an Embedded Controller (EC). System Management Services could be implemented on-chip on same SoC as the PE, off-chip on an external chip or chipset, or through a combination of both.

I Physical interfaces used to present System Management Services to the PE are implementation defined. Implementations may choose appropriate interfaces, as long as they can be abstracted to operating systems via standard firmware mechanisms.

1.4 Future requirements

The following section lists a preview of new rules that will be required in a future PC-BSA Level, which will be published in a future version of this document.

The list of rules to be implemented for future PC-BSA Level is available in Section 2.2.

1.4.1 Watchdog

R_P_L2WD_01 The base PC system must implement a Non-secure Generic watchdog as specified by B_WD_01, B_WD_02, B_WD_03, B_WD_04, and B_WD_05 in Arm BSA [3].

2 PC-BSA checklist

This section lists the minimum hardware requirements required to install, boot, and run a PC operating system on bare-metal or within a virtualization environment.

Note

The list of “Related rules from other specifications” below is indicative, and provided only for reference.

2.1 PC-BSA level 1 checklist

Module	Required Rule ID	Related rules from other specifications (for reference only)
Base	P_L1_01	
PE	P_L1PE_01	S_L3PE_01 (SBSA)
PE	P_L1PE_02	S_L3PE_02 (SBSA)
PE	P_L1PE_03	S_L3PE_03 (SBSA)
PE	P_L1PE_04	B_PE_25 (BSA)
PE	P_L1PE_05	S_L3PE_04 (SBSA)
PE	P_L1PE_06	S_L4PE_02 (SBSA)
PE	P_L1PE_07	S_L4PE_03 (SBSA)
PE	P_L1PE_08	S_L4PE_04 (SBSA)
Memory	P_L1MM_01	S_L3MM_01 (SBSA), S_L3MM_02 (SBSA)
Interrupt	P_L1GI_01	S_L3GI_01 (SBSA)
Interrupt	P_L1GI_02	S_L3GI_02 (SBSA)
Interrupt	P_L1GI_03	
Interrupt	P_L1GI_04	
Interrupt	P_L1PP_01	S_L3PP_01 (SBSA)
SMMU	P_L1SM_01	S_L3SM_01 (SBSA)
SMMU	P_L1SM_02	S_L4SM_01 (SBSA)
SMMU	P_L1SM_03	S_L4SM_02 (SBSA)
SMMU	P_L1SM_04	S_L4SM_03 (SBSA)
SMMU	P_L1SM_05	S_L7SM_01 (SBSA)
PCIe	P_L1PCI_1	S_L4PCI_1 (SBSA)
PCIe	P_L1PCI_2	S_L4PCI_2 (SBSA)
NV Store	P_L1NV_01	
Security	P_L1SE_01	
Security	P_L1SE_02	

Module	Required Rule ID	Related rules from other specifications (for reference only)
Security	P_L1SE_03	
Security	P_L1SE_04	
Security	P_L1SE_05	
TPM	P_L1TP_01	
TPM	P_L1TP_02	
TPM	P_L1TP_03	
TPM	P_L1TP_04	

2.2 Future Level checklist

In addition to the PC-BSA Level 1 rules in Section 2.2, the following additional rules are required.

Module	Required Rule ID	Related rules from other specifications (for reference only)
Watchdog	P_L2WD_01	S_L3WD_01 (SBSA)