Arm® Server Base System Architecture 6.1

Platform Design Document

Non-confidential

Document number: DEN0029E

Contents

Release information

Version 6.1 (15 Sep 2020)

- The document is reorganized to be a supplement of the Arm BSA document.

Version 6.0 (16 Sep 2019)

- Armv8.5 requirements (115).
- Instruction to data to instruction coherency (175).
- Nanosecond units for system counter (159).
- Rules for SMMU to ease page sharing between PE and SMMUs (158).
- MSI(-X) must be mapped to LPI (173).
- Mention ServerReady and GIC level-3 clarification (166).
- Clarification on this meaning MSI(-X) (103).
- Typo in word address, changed to addresses (102).
- mislabelling of UARTIMSC in table 15 (101).
- IO virtualization clarifications (112).
- Armv8 RAS extension requirements (133).
- Relaxation of SVE requirement (162).
- Errata clarification on level 5 interrupt controller section of SBSA5.0B (104).
- Level 5 - Section 4.4.2 Interrupt Controller(117).
- SMMU and HTTU support (134).
- SBSA Typo on Level 3 PE requirements for SVE (152).
- PCI Enumeration related requirements (110).
- PCIe RCiEP and iEP related requirements (108).
- MSI support in SMMU for events (194).

Version 5.0 (30 May 2018)

- RAS requirements (5).
- ROP and JOB requirements for SBSA (6).
- SBSA and SVE (7).
- Nested virtualization and SBSA (8).
- MPAM requirements for SBSA (9).
- Invisible caches (ban type 2 caches) AKA Forced WB (10).
- Add Activity Monitor Requirements to SBSA (11).
- Crypto Requirements to SBSA (12).
- Add support for 48-bit operating systems booting on Armv8.2+ systems with 52-bit PA (13).
- Ban non-standard interrupt controller (14).
- Base frequency standardisation (15).
- Assign PPIs for new timers in v8.4 (16).
- Secure EL2 not required (17).
- TLBI range homogeneity (18).
- SVE heterogeneity (19).
- PCIe clarifications (21).
- UART clarifications (22).
- PCIe requirements for assignable devices (23).
- PTM for SBSA-based systems (24).
- PCIe deadlocks (25).
- ACS and Peer-to-peer (26).
- TPM guidance for SBSA (27).
- Deprecate Levels 0, 1, and 2 (28).
- Errata Remove references to Prince Algorithm (29).

- Errata UART Trigger Levels (30).
- Heterogeneity in Server (31).
- Clean to point of persistence support (33).
- Watchdog scaling in SBSA (36).
- Clarify that PEs must be able to accesses all Non-secure address space (37).
- Appendix I needs to be clearer about RID spaces do not strictly need to supply all 16 bits of width (38).
- Address space input into an SMMU from a device is entirely contiguous, no holes (39).
- Clarify the uniqueness requirements of SPIs that are used to handle legacy PCIe interrupts (A/B/C/D) (40).
- SBSA Armv8.4 and SMMUv3.2 rules on break before make for page tables (41).
- Add reservation for trace buffer overflow PPI (42).
- FP16 support in SBSA (43).
- Correction on stating devices describe their ability to be virtualised through FW (44).
- Clarify IO BAR usage (45).
- Generalise SMMU requirements (46).
- Relaxations in future GIC PPI reservation for Level 5 (79).

Version 3.0 (01 Feb 2016)

- Initial Release. Non-confidential.

Version 3.1 (01 Feb 2016)

- Change of Proprietary Notice. Addition of release history. Non-confidential.

Arm Non-Confidential Document Licence (“Licence”)

About this document

Terms and abbreviations

Term	Meaning
ACS	Access Control Services. A set of features intended to ensure that uncontrolled peer-to-peer transaction cannot occur. See PCIe specification [1] for more details.
AER	Advanced Error Reporting. A PCIe feature that enables software to isolate and analyze errors with fine granularity. See PCIe specification [1] for more details.
ARE	Affinity Routing Enable (GICv3 [2]).
Arm ARM	Arm Architecture Reference Manual; see [3].
Arm recommends	This phrase is used in statements when Arm **suggests** that the implementer do something, but the decision remains with the implementer.
Arm strongly recommends	This phrase is used in statements when Arm **expects** that the implementer do something, but the decision remains with the implementer.
ATS	Address Translation Services
Base Server System	A system that is compliant with the Server Base System Architecture.
ECAM	Enhanced Configuration Access Mechanism
GIC	Generic Interrupt Controller
I/O coherent	A device is I/O coherent with the PE caches if its transactions snoop the PE caches for cacheable regions of memory. The PE does not snoop the device's cache.
LPI	Locality-specific Peripheral Interrupt (GICv3 [2]).
MMIO	Memory Mapped Input Output
P2P or Peer-to-peer	Traffic that is sent directly between two PCIe endpoints. See PCIe specification [1] for more details.
PCI Host Bridge (PHB)	A host-to-PCI bridge; this provides a design-time fixed range of bus and memory/IO resource ranges to the system. A PHB connects a set of root ports to the host.
PE	Processing Element, as defined in the Arm Architecture Reference Manual. Usually a single hardware thread of a PE.
PMU	Performance Monitoring Unit
PPI	Private Peripheral Interrupt
PRI	Page Request Interface
PTM	Precision Time Measurement. PCIe standard specified method for finding the relationship between a PTM master clock and another clock in a device in the same hierarchy.
RCEC	Root Complex Event Collector
RCiEP	Root Complex integrated End Point

Term	Meaning
Root Complex (RC)	A collection of PHBs and therefore RPs. A RC might be conceptually composed of one or more PCIe Host Bridge (PHB). A RC provides a PCI segment. All PHBs in a RC are on the same segment.
Root Port (RP)	A Root Port is associated with a physical port and appears in PCIe config space as a PCI bridge with a type 1 header, A RP is the root of a PCIe hierarchy.
SBBR	Server Base Boot Requirements [4].
SBSA	Server Base System Architecture.
SGI	Software-Generated Interrupt
SPI	Shared Peripheral Interrupt
SR-IOV	Single Root I/O virtualization. This is a method that allows a PCIe device to be virtualized. See PCIe specification [1] for more details.
SRE	System Register interface Enable (GICv3 [2]).
System firmware data	System description data structures, for example ACPI or Flattened Device Tree.
TCG	Trusted Computing Group
TPM	Trusted Platform Module. TPM is a technology, typically implemented through a secure microcontroller, that can securely store artifacts that are 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] *PCI Express Base Specification Revision 5.0, version 1.0*. PCI-SIG.

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

[3] *DDI 0487 Arm® Architecture Reference Manual ARMv8, for the ARMv8-A architecture profile*. Arm Ltd.

[4] *DEN 0044 Arm Base Boot Requirements*. Arm Ltd.

[5] *DEN 0094A Arm® Base System Architecture version 1.0*. Arm Ltd.

[6] *DEN 0029C Server Base System Architecture Version 6.0*. Arm Ltd.

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

[8] *DEN 0068 CoreSight Base System Architecture*. Arm Ltd.

[9] *DDI 0587 Arm Reliability, Availability and Serviceability (RAS) specification Armv8, for the Armv8-A architecture profile*. Arm Ltd.

Rules-based writing

This specification consists of a set of individual rules. Each rule is clearly identified by the letter R.

Rules must not be read in isolation, and where more than one rule relating to a particular feature exists, individual rules are grouped into sections and subsections to provide the proper context. Where appropriate, these sections contain a short introduction to aid the reader. An implementation which is compliant with the architecture must conform to all of the rules in this specification.

Some architecture rules are accompanied by rationale statements which explain why the architecture was specified as it was. Rationale statements are identified by the letter X.

Some sections contain additional information and guidance that do not constitute rules. This information and guidance is provided purely as an aid to understanding the architecture. Information statements are clearly identified by the letter I.

Implementation notes are identified by the letter U.

Software usage descriptions are identified by the letter S.

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

Rules, rationale statements, information statements, implementation notes and software usage statements are collectively referred to as *content items*.

Identifiers

Each content item may have an associated identifier which is unique within the context of this specification.

When the document is prior to beta status:

- Content items are assigned numerical identifiers, in ascending order through the document (*0001*, *0002*, . . .).
- Identifiers are volatile: the identifier for a given content item may change between versions of the document.

After the document reaches beta status:

- Content items are assigned random alphabetical identifiers (*HJQS*, *PZWL*, . . .).
- Identifiers are preserved: a given content item has the same identifier across versions of the document.

Examples

Below are examples showing the appearance of each type of content item.

R This is a rule statement.

$\mathtt{R}_{\mathtt{X001}}$ This is a rule statement.

I This is an information statement.

X This is a rationale statement.

U This is an implementation note.

S This is a software usage description.

Feedback

Arm welcomes feedback on its documentation.

If you have comments on the content of this manual, send an e-mail to errata@arm.com. Give:

- The title (Server Base System Architecture).
- The document ID and version (DEN0029E 6.1).
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

1 Server Base System Architecture

1.1 Arm Base System Architecture

This document is a supplement to the Arm Base system Architecture document version 1.0 [5].

1.2 Introduction

Server Base System Architecture (SBSA) specifies a hardware system architecture, based on Arm 64-bit architecture, that server system software, for example operating systems, hypervisors, and firmware can rely on. SBSA extends the requirements specified in the Arm BSA [5].

The Server Base System Architecture embeds the notion of levels of functionality. Each level adds functionality over and above a previous level. Unless explicitly stated, all specification items that belong to level N apply to levels greater than N.

All views of a level are required to be implemented to be compliant to a level.

SBSA Compliance is a requirement for the Arm ServerReady program.

An implementation is consistent with a level of the Server 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 can be exploited by software without unexpectedly poor performance.

The Server Base Boot Requirements (SBBR) specification [4] describes firmware requirements for an Arm server system. Where this specification refers to system firmware data, it refers to firmware specified in the SBBR.

1.2.1 SBSA levels and minimum component versions

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

This table is indicative only. The rules in each level describe the specific features that are required to be compliant to that level.

Table 2: SBSA level mapping summary

Level	PE: A profile	SMMU	GIC
3	v8.0	v2 or v3	v3.0
4	v8.3	v3.0	v3.0
5	v8.4	v3.2	v3.0
6	v8.5	v3.2	v3.0

Note

Each SBSA levels requires a set of PE features. These features were introduced in different revisions of the architecture. Generally, features that are available in version X of the architecture extension can be optionally implemented in version X-1. Some features can be backported even further. Please refer to extension documents for more details. Table 2 indicates the version of the architecture extension at which the required features were introduced.

Note

The levels between Level 3 and Level 6 that are presented in this document are semantically equivalent to the corresponding levels in SBSA 6.0 [6].

1.3 Level 3

$\mathrm{R}_{\mathrm{S\_L3\_01}}$ The rules from Arm BSA, [5] as listed in Section 1.7.1 must be implemented.

I The PE Security requirements in Arm BSA [5] are relaxed to become a recommendation. This is to maintain consistency with systems which are certified as ServerReady v1.0.

1.3.1 PE architecture

$\mathrm{R}_{\mathrm{S\_L3PE\_01}}$ PEs must support 4KB and 64KB translation granules at stage 1 and stage 2.

$\mathrm{R}_{\mathrm{S\_L3PE\_02}}$ PEs must implement 16-bit ASID support.

$\mathrm{R}_{\mathrm{S\_L3PE\_03}}$ PEs must implement AArch64 at all implemented Exception levels.

I Level 3 systems can be implemented using the Armv8.0 revision of the architecture, or higher. FEAT_LPA [3], 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 granule.

$\mathrm{R}_{\mathrm{S\_L3PE\_04}}$ Server base systems that make use of FEAT_LPA must provide a mode where the memory map is wholly contained inside 2^{48} bytes (256TB). This is required to support operating systems that do not use the 64KB granule.

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

1.3.2 Memory map

$\mathrm{R}_{\mathrm{S\_L3MM\_01}}$ To enable Non-secure EL2 hypervisors to use a 64KB translation granule at stage 2 MMU translation, the base system must ensure that all memory and peripherals can be mapped using 64KB stage 2 pages and must not require the use of 4KB pages at stage 2.

$\mathrm{R}_{\mathrm{S\_L3MM\_02}}$ Peripherals that will be assigned to different virtual machines will be situated within different 64KB regions of memory.

1.3.3 Interrupt controller

$\mathrm{R}_{\mathrm{S\_L3G\_01}}$ A base server system must implement an interrupt controller that is compliant with the GICv3 or higher architecture [2].

$\mathrm{R}_{\mathrm{S\_L3G\_02}}$ All MSI and MSI-X targeting hypervisor and operating system software must be mapped to LPI.

1.3.4 PPI assignments

$R_{S\_L3PP\_01}$ The Interrupt IDs must be the same as the recommended values mentioned in Arm BSA [5].

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.

$R_{S\_L3SM\_01}$ If PEs that are used by the base system support TLB range instructions, then all OS visible masters that contain a TLB must support range invalidates. See FEAT_TLBIRANGE in [3].

1.3.6 Watchdog

$R_{S\_L3WD\_01}$ The server base system must implement a Non-secure Generic watchdog as described in Watchdogs section in Arm BSA [5].

1.4 Level 4

The list of rules to be implemented for Level 4 is available in Section 1.7.2.

1.4.1 PE architecture

In addition to the level 3 requirements, the following must be true of the PEs in the base server system:

$R_{S\_L4PE\_01}$ All PEs must implement the RAS extension that is introduced in Armv8.2. See FEAT_RAS in [3].

$R_{S\_L4PE\_02}$ 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.

$R_{S\_L4PE\_03}$ All PEs must implement FEAT_VMID16.

$R_{S\_L4PE\_04}$ All PEs must implement FEAT_VHE.

1.4.2 System MMU and device assignment

$R_{S\_L4SM\_01}$ Stage 1 System MMU functionality must be provided by a System MMU that is compatible with the Arm SMMUv3, or higher, architecture revision.

$R_{S\_L4SM\_02}$ Stage 2 System MMU functionality must be provided by a System MMU that is compatible with the Arm SMMUv3, or higher, architecture revision.

$R_{S\_L4SM\_03}$ The integration of the System MMUs is compliant with the rules in SMMUv3 integration appendix in Arm BSA [5].

1.4.3 Peripheral subsystems

$R_{S\_L4PCI\_1}$ All peripherals that are intended for assignment to a virtual machine or a user space device driver must be based on PCI Express.

$R_{S\_L4PCI\_2}$ There must be no OS observable use of PCIe Enhanced Allocation [7].

1.5 Level 5

The list of rules to be implemented for level 5 is available in Section 1.7.3.

1.5.1 PE architecture

In addition to the level 4 requirements, the following must be true of the PEs in the base server system:

R_{S_L5PE_01} All PEs must support changing of translation table mapping size (FEAT_BBM) using the Level 1 or Level 2 solution that is proposed in the Armv8.4 extension. Arm recommends Level 2. See Section 1.5.3 for the equivalent requirements for the SMMU.

R_{S_L5PE_02} All PEs must implement pointer authentication using the standard algorithm that is defined by the Arm architecture [3], as indicated by ID_AA64ISAR1_EL1[7:4]==0001. See FEAT_PAuth in [3].

R_{S_L5PE_03} PEs that are based on Armv8.4 must implement the requirements of the CS-BSA combination C [8].

R_{S_L5PE_04} All PEs must implement the Activity Monitors Extension (AMU).

R_{S_L5PE_05} Where export control allows, all PEs must implement cryptography support for SHA3 and SHA512. See FEAT_SHA3 and FEAT_SHA512 in [3].

Note

Arm recommends that FEAT_SM3 and FEAT_SM4 is also supported in hardware aimed at the Chinese market.

R_{S_MPAM_PE} Implementation of the MPAM extension (FEAT_MPAM) is OPTIONAL, however if implemented, the following minimal requirements must be met by the implementation:

- provide a minimum of 16 physical partition IDs.
- provide virtualization support with a minimal of 8 virtual partition IDs.
- provide a minimal of 2 performance monitor groups.
- provide cache portion partitioning in the last level cache.

I Last level cache refers to an on SoC cache, as opposed to an off chip DRAM cache.

Note

Arm strongly recommends that the number of partition IDs scales with the number of PEs.

R_{S_L5PE_06} All PEs must provide support for stage 2 control of memory types and cacheability, as introduced by the Armv8.4 extensions. See FEAT_S2FWB in [3].

R_{S_L5PE_07} All PEs must implement enhanced nested virtualization, that is provided by HCR_EL2.NV2 and the VNCR_EL2 register. See FEAT_NV2 in [3].

1.5.2 Interrupt controller

R_{S_L5GI_01} Only a GIC v3, or higher, interrupt controller will be visible to operating system software. Other forms of interrupt controller, for example interrupt combining or forwarding engines, are not permissible, if they require a platform specific kernel driver.

R_{S_L5GI_02}

Only a GIC v3, or higher, interrupt controller will be visible to the hypervisor. Other forms of interrupt controller, for example interrupt combining or forwarding engines, are not permissible, if they require a platform specific kernel driver.

1.5.3 System MMU and device assignment

$R_{S\_L5SM\_01}$ SMMU implementations must be compliant with the Arm SMMUv3.2 architecture revision or higher.

$R_{S\_L5SM\_02}$ SMMU implementations must provide level 1 or level 2 support for translation table resizing.

Note

Arm recommends the SMMU implements Level 2. If the SMMU implementation provides Level 2, then Arm recommends that the PE also provides level 2.

X MPAM architecture requires that all masters that can access an MPAM controlled resource, must support passing MPAM ID information.

1.5.4 Clock and Timer subsystem

1.5.4.1 Operating system

$R_{S\_L5TI\_01}$ A system that is compatible with level 5 will implement a generic counter which counts in nanosecond units. This means that, to the operating system, the reported frequency will be 1GHz.

I Systems are permitted to use the counter scaling (FEAT_CNTSC) that is introduced in Armv8.4 as a method to implement this.

Note

Arm strongly recommends that this type of system uses revision 1 of the generic watchdog. This is because at 1Ghz the watchdog timeout refresh period is limited to just over 4s.

1.5.5 Watchdog

I Arm strongly recommends that SBSA Level 5 systems use revision 1 of the generic watchdog. This is because at 1Ghz the watchdog timeout refresh period is limited to just over 4s.

1.5.6 PPI assignments

$R_{S\_L5PP\_01}$ In addition to the PPI assignment specified in Arm BSA [5], the following PPIs are reserved by the SBSA specification:

Table 3: PPI assignments

Interrupt ID	Interrupt	Description
1056-1071	Reserved	Reserved for future SBSA usage.

Interrupt ID	Interrupt	Description
1088-1103	Reserved	Reserved for future SBSA usage.

1.6 Level 6

The list of rules to be implemented for Level 6 is available in Section 1.7.4.

1.6.1 Level 6 PE requirements

In addition to the Level 5 requirements, the following must be true of the PEs in the base server system:

RS_L6PE_01 PE security requirements as described in [5] are mandatory.

RS_L6PE_02 PEs must provide support for Branch Target Identification. Support is indicated by register ID_AA64PFR1_EL1.BT== b0001. See FEAT_BTI in [3].

RS_L6PE_03 PEs must protect against timing faults being used to guess translation table mappings by implementing the TCR_EL1.E0PD0 and TCR_EL1.E0PD1 controls, and the same in TCR_EL2. See FEAT_E0PD in [3]. Support is indicated by ID register ID_AA64MMFR2_EL1.E0PD==b0001.

RS_L6PE_04 All PEs must implement FEAT_PMUv3p5 [3].

RS_L6PE_05 Hardware updates to Access flag and Dirty state in translation tables, as indicated by ID_AA64MMFR1_EL1. HAFDBS = 0b0010, must be supported. See FEAT_HAFDBS in [3].

Note

Arm strongly recommends that, the server base system removes the need for data cache clean for instruction to data coherency, and instruction invalidation for instruction to data coherency, as indicated by CTR_EL0.IDC == 0b1 and CTR_EL0.DIC == 0b1 respectively.

RS_L6PE_06 PEs must provide support for enhanced virtualization traps as indicated by ID_AA64MMFR2_EL1.EVT==b0010. See FEAT_EVT in [3].

1.6.2 System MMU and device assignment

RS_L6SM_01 The SMMU must implement coherent access to in memory structures, queues, and page tables as indicated by SMMU_IDR0.COHACC = 0b1.

RS_L6SM_02 The SMMU must support hardware translation table update (HTTU) of the Access flag and the Dirty state of the page for AArch64 translation tables, as indicated by SMMU_IDR0.HTTU = 0b10.

RS_L6SM_03 The SMMU must support Message Signaled Interrupts as indicated by SMMU_IDR0.MSI = 0b1.

1.6.3 Watchdog

RS_L6WD_01 The generic watchdog revision must be v1, W_IIDR==0001b.

X In Generic watchdog v0, at 1 GHz the watchdog timeout refresh period is limited to just over 4 seconds.

1.6.4 Armv8 RAS extension requirements

$R_{S\_RAS\_01}$ PEs or other system components that implement the Armv8 RAS extension [9] must:

- Use Private Peripheral Interrupts for ERI or FHI if the only interface available for a RAS node is System register based.
- Use the generic counter time base if the timestamp extension is implemented. This means that ERR<n>FR.TS must be either b00 or b01.

$R_{S\_RAS\_02}$ Support for error injection is OPTIONAL, however Arm recommends that if error injection is supported, the standard programming model that is described in the Armv8 RAS extension version 1.1 is followed. Support for error injection is indicated by ERR<n>FR.INJ==b01.

Note

Arm recommends that the Timestamp error extension is implemented.

1.7 SBSA checklist

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

All rule IDs which begin with 'B' are defined in the Arm BSA document [5].

1.7.1 SBSA Level 3 checklist

PE	Rule ID
Operating system	
	B_PE_01
	B_PE_02
	B_PE_03
	B_PE_04
	B_PE_05
	B_PE_06
	B_PE_07
	B_PE_08
	B_PE_09
	B_PE_10
	B_PE_11
	B_PE_12
	B_PE_13
	B_PE_14

PE	Rule ID
	S_L3PE_01
	S_L3PE_02
	S_L3PE_03
	S_L3PE_04
Hypervisor	
	B_PE_18
	B_PE_19
	B_PE_20
	B_PE_21
	B_PE_22
Platform security	
	B_PE_23
	B_PE_24

Memory map	Rule ID
Operating system	
	B_MEM_01
	B_MEM_02
	B_MEM_03
	B_MEM_04
	B_MEM_05
	B_MEM_06
	B_MEM_07
	S_L3MM_01
	S_L3MM_02
Platform security	
	B_MEM_08
	B_MEM_09

Interrupts	Rule ID
Operating system	
	S_L3G_01
	S_L3G_02

Interrupts	Rule ID
	B_GIC_03
	B_GIC_04
	B_GIC_05
Operating system	
	S_L3PP_01
	B_PPI_01
Hypervisor	
	B_PPI_02
Platform security	
	B_PPI_03

SMMU	Rule ID
Operating system	
	B_SMMU_01
	B_SMMU_02
	B_SMMU_06
	B_SMMU_07
	B_SMMU_08
	B_SMMU_12
	S_L3SM_01
Hypervisor	
	B_SMMU_16
	B_SMMU_17
	B_SMMU_18
	B_SMMU_19
	B_SMMU_21

Timer subsystem	Rule ID
Operating system	
	B_TIME_01
	B_TIME_02
	B_TIME_03
	B_TIME_04

Timer subsystem	Rule ID
	B_TIME_05
	B_TIME_06
	B_TIME_07
	B_TIME_08
	B_TIME_09
	B_TIME_10

Power and wakeup	Rule ID
	B_WAK_01
	B_WAK_02
	B_WAK_03
	B_WAK_04
	B_WAK_05
	B_WAK_06
	B_WAK_07
	B_WAK_08
	B_WAK_09
	B_WAK_10
	B_WAK_11

Peripherals	Rule ID
Operating system	
	B_PER_01
	B_PER_02
	B_PER_03
	B_PER_04
	B_PER_05
	B_PER_06
	B_PER_07
	B_PER_08
	B_PER_09
	B_PER_10
	B_PER_12

Peripherals	Rule ID
Platform security	
	B_PER_11

Watchdog	Rule ID
	B_WD_01
	B_WD_02
	B_WD_03
	B_WD_04
	B_WD_05
	B_WD_06

1.7.2 SBSA Level 4 checklist

In addition to the SBSA Level 3 rules in Section 1.7.1, the following additional rules are required.

Module	Rule ID
PE	S_L4PE_01
PE	S_L4PE_02
PE	S_L4PE_03
PE	S_L4PE_04
SMMU	~~B_SMMU_08~~
SMMU	S_L4SM_01
SMMU	S_L4SM_02
SMMU	S_L4SM_03
PCIe	S_L4PCI_1
PCIe	S_L4PCI_2

1.7.3 SBSA Level 5 checklist

In addition to the SBSA Level 4 rules in Section 1.7.2, the following additional rules are required.

Module	Rule ID
PE	S_L5PE_01
PE	S_L5PE_02
PE	S_L5PE_03

Module	Rule ID
PE	S_L5PE_04
PE	S_L5PE_05
PE	S_L5PE_06
PE	S_L5PE_07
PE	S_MPAM_PE
GIC	S_L5GI_01
GIC	S_L5GI_02
SMMU	~~S_L4SM_01~~
SMMU	S_L5SM_01
SMMU	S_L5SM_02
SMMU	S_L5SM_03
SMMU	S_L5SM_04
SMMU	B_SMMU_09
SMMU	B_SMMU_11
SMMU	B_SMMU_20
SMMU	B_SMMU_22
Timer	S_L5TI_01
Interrupt	S_L5PP_01

1.7.4 SBSA Level 6 checklist

In addition to the SBSA Level 5 rules in Section 1.7.3, the following additional rules are required.

Module	Rule ID
PE	B_PE_16
PE	B_PE_17
PE	S_L6PE_01
PE	S_L6PE_02
PE	S_L6PE_03
PE	S_L6PE_04
PE	S_L6PE_05
PE	S_L6PE_06
PE	B_SEC_01
PE	B_SEC_02
PE	B_SEC_03
PE	B_SEC_04

Module	Rule ID
PE	B_SEC_05
SMMU	B_SMMU_03
SMMU	B_SMMU_04
SMMU	B_SMMU_05
SMMU	B_SMMU_13
SMMU	B_SMMU_14
SMMU	B_SMMU_15
SMMU	B_SMMU_23
SMMU	S_L6SM_01
SMMU	S_L6SM_02
SMMU	S_L6SM_03
Watchdog	S_L6WD_01
RAS	S_RAS_01
PCI	B_REP_1
PCI	B_IEP_1

A Level 3 - firmware

Level 3 - firmware is an optional additional set of requirements for a Level 3 system. Level 3 - firmware is designed to give a base set of functionality that standard platform firmware can rely on. A system that is compliant with level 3 and not compliant with level 3 - firmware is still a fully compliant level 3 system. The system has all the features required by the operating systems and hypervisors.

A.1 Memory map

The system must provide some memory mapped in the Secure address space. The memory must not be aliased in the Non-secure address space. The amount of Secure memory provided is platform-specific as the intended use of the memory is for platform-specific firmware.

All Non-secure on-chip masters in a base server system that are expected to be used by the platform firmware must be capable of addressing all of the Non-secure address space. If the master goes through a SMMU then the master must be capable of addressing all of the Non-secure address space even when the SMMU is off.

A.2 Clock and Timer Subsystem

A system compatible with level 3 - firmware must also include a Secure wakeup timer in the form of the memory mapped timer described in the Armv8 ARM [3] This timer must be mapped into the Secure address space, and the timer expiry interrupt must be presented to the GIC as an SPI. This timer is called the Secure system wakeup timer.

The Secure wakeup timer does not require a virtual timer to be implemented. The virtual offset register can Read-As-Zero, where writes to the virtual offset register in CNTCTLBase frame are ignored. The timer is not required to have a CNTEL0Base frame.

The following table summarizes which address space the register frames related to the Secure wakeup timer should be mapped on to.

Register Frame	
CNTControlBase	Secure
CNTReadBase	Not required
CNTCTLBase	Secure
CNTBaseN	Secure

CNTCTLBase might be shared among multiple timers, including various Secure and Non-secure timers. The SBSA specification does not require this.

Note

- GICv3 does not support Secure LPI. Therefore, the Secure system timer interrupt must not be delivered as LPI.
- It is recognized that in a large system, a shared resource like the system wakeup timer can create a

system bottleneck. This is because, access to it must be arbitrated through a system-wide lock. Level 3-firmware requires just a single timer so that standard firmware implementations have a guaranteed timer resource across platforms. We anticipate that large PE systems will implement a more scalable solution like one timer for each PE.

A.3 Watchdog

The required behavior of watchdog signal 1 of the Non-secure watchdog is modified in level 3 - firmware and is required to be routed as an SPI to the GIC. It is expected that this SPI be configured as an EL3 interrupt, directly targeting a single PE.

A system compatible with level 3- firmware must implement a second watchdog, and is referred to as the Secure watchdog. It must have both its register frames mapped in the Secure memory address space and must not be aliased to the Non-secure address space.

Watchdog Signal 0 of the Secure watchdog must be routed as an SPI to the GIC and it is expected this will be configured as an EL3 interrupt, directly targeting a single PE.

Note

- GICv3 does not support Secure LPI. The Secure watchdog interrupts must not be delivered as LPI.
- Only directly-targeted SPIs are required to wake a PE. Programming the watchdog SPI to be directly targeted ensures delivery of the interrupt independent of PE power states. However, it is possible to use a 1 of N SPI to deliver the interrupt, if one of the target PEs is running. See Arm BSA [5] for information about SPI waking a PE.

Watchdog Signal 1 of the Secure watchdog must be routed to the platform. In this context, platform means any entity that is more privileged than the code running at EL3. Examples of the *platform* component that services Watchdog Signal 1 are a system control processor, or dedicated reset control hardware.

The action that is taken on the raising of Watchdog Signal 1 of the Secure watchdog is platform specific.