Arm® BSA Architecture Compliance

Revision: r1p0

User Guide

Non-Confidential

Issue 04

102504_0100_04_en

Arm® BSA Architecture Compliance

User Guide

Release Information

Document history

Issue	Date	Confidentiality	Change
0005-01	12 May 2021	Non-Confidential	Alpha release
0009-02	26 July 2021	Non-Confidential	Beta release
0100-01	6 September 2021	Non-Confidential	REL v1.0
0100-02	29 October 2022	Non-Confidential	REL v1.0.1
0100-03	28 March 2023	Non-Confidential	REL v1.0.4
0100-04	28 September 2023	Non-Confidential	REL v1.0.6

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com.

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

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

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction

1.1 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Convention	Use
italic	Citations.
bold	Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

Caution

Recommendations. Not following these recommendations might lead to system failure or damage.

Warning

Requirements for the system. Not following these requirements might result in system failure or damage.

Danger

Requirements for the system. Not following these requirements will result in system failure or damage.

Note

An important piece of information that needs your attention.

Tip

A useful tip that might make it easier, better or faster to perform a task.

Remember

A reminder of something important that relates to the information you are reading.

1.2 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
GICv3 and GICv4 Software Overview	DAI0492	Non-Confidential
Arm® Base System Architecture 1.0	DEN0094C	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
Arm® Architecture Reference Manual for A-profile architecture	DDI0487	Non-Confidential
Arm® Generic Interrupt Controller Architecture Specification for GIC architecture version 3.0 and version 4.0	IHI0069	Non-Confidential

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com.

1.3 Other information

See the Arm® website for other relevant information.

- Arm® Developer.
- Arm® Documentation.
- Technical Support.
- Arm® Glossary.

2. Overview

This chapter provides an overview of the BSA tests and the test IDs.

2.1 Abbreviations

This section lists the abbreviations used in this document.

Table 2-1: Abbreviations and expansions

Abbreviation	Expansion
ACPI	Advanced Configuration and Power Interface
BSA	Base System Architecture
DT	Device Tree
GIC	Generic Interrupt Controller
HVC	HyperVisor Call
PAL	Platform Abstraction Layer
PCIe	Peripheral Component Interconnect express
PE	Processing Element
PSCI	Power State Coordination
SMC	Secure Monitor Call
SMMU	System Memory Management Unit
UEFI	Unified Extensible Firmware Interface

2.2 Overview of tests

The following table describes the general divisions of Base System Architecture (BSA) tests between Unified Extensible Firmware Interface (UEFI) shell application and Linux application.

Table 2-2: Test environment and modules

Test environment	Modules
UEFI shell	PE, GIC, Timers, Watchdog, Wakeup and Power, PCIe, Memory map, Exerciser, Peripheral, and SMMU
Linux command line	PCIe, Memory map, and Peripheral

2.3 Test IDs

This section provides information on module names and module IDs.

The test ID of each test is generated as an addition to the module ID and unit test ID. For a given module, the unit test ID begins from 1.

The following table lists the module name and module IDs.

Table 2-3: Module name and Module ID

Module name	Module ID
PE	0
Memory Map	100
GIC	200
SMMU	300
Timer	400
Wakeup and Power	500
Peripheral	600
Watchdog	700
PCIe	800
Exerciser	900

Note

Each module has tests classified as operating system, hypervisor, and platform security as defined by the BSA v1.0 (C) specification.

3. UEFI shell application

This chapter provides information on executing tests from the UEFI shell application and the PAL API implementation.

3.1 UEFI shell application arguments

This section provides information on the UEFI shell application arguments.

Run the UEFI shell application with the following set of arguments.

```
uefi shell> bsa.efi [-v <n>] [-skip <x,y,z, ..>] [-f <file name>] [-os] [-hyp] [-ps]
 [-dtb <file name>]
-t <x,y,z> [-m <x,y,z>] [-mmio] [-sbsa] [-timeout <wakeup test timeout multiple>] [-
p2p] [-cache] [-el1physkip]
```

Note

The shell session becomes unusable after all the BSA tests are run and the test results are printed on the UEFI console.

The following table provides descriptions to the arguments.

Table 3-1: Description of UEFI application arguments

Argument	Description
`-v`	Print level **1** INFO and above. **2** DEBUG and above. **3** TEST and above. **4** WARN and ERROR. **5** ERROR.
`-skip`	Overrides the suite to skip the execution of a particular test(s) or/and module(s). For example, 302 skips test case with ID = 302. 200 skips all tests in module with ID = 200. For more information on module IDs, see the 2.3 Test IDs on page 9.
`-f`	File name to which the output log is written.

`-os`
`-hyp`
`-ps`
	By default, all the operating system, hypervisor, and platform security view tests are run.
	To run specific tests, add the following options:
	`-os` Run the operating system view tests.
	`-hyp` Run the hypervisor view tests.
	`-ps` Run the platform security view tests.

Argument	Description
`-dtb`	Dumps the board Device Tree (DT) blob into the specified file.
`-t`	To run only multiple selected tests.
`-m`	To run only multiple selected modules. **Note:** -m will override -t if used on the same module.
`-sbsa`	Flag to pass to run bsa tests as per SBSA requirements.
`-timeout`	Timeout value for wakeup test.
`-p2p`	Pass this flag to indicate system support PCIe p2p.
`-cache`	Pass this flag to indicate system support PCIe address translation cache.
`-mmio`	Enables all the mmio read or write prints. **Note:** • To enable pal_mmio_read or write prints, use with -v 1. • Enables prints from specific module by using module id. For example, -mmio 200, enables for GIC module, and -mmio 0, enables for PE module.
`-el1physkip`	Skips EL1 register checks.

Note

`-dtb` option is for platforms that present DT files only.

Examples

The following examples show how to run the UEFI shell application using arguments:

```
shell > bsa.efi -v 2 -skip 200,302 -f acs.txt -os -dtb platform.dtb
```

The set of parameters shown in the code block:

- Prints messages with verbosity of 2 and above.
- Tests for compliance against operating system view tests.
- Skips execution of all tests belonging to Generic Interrupt Controller (GIC) module and test number 302.
- Stores the log messages to the `acs.txt` file.
- Saves the firmware DT into the `platform.dtb` file.

```
shell > bsa.efi -m 200 -skip 202
```

The set of parameters shown in the code block:

- Runs only GIC module.
- Skips GIC test 202.

3.2 UEFI shell implementation of PAL APIs

This section provides information on infrastructure APIs and module-specific APIs.

Booting to a UEFI shell is a prerequisite for running a BSA test.

Infrastructure APIs

The following table describes the Platform Abstraction Layer (PAL) APIs and UEFI interfaces.

Table 3-2: PAL APIs and UEFI interfaces

PAL API	UEFI interface
`pal_print`	AsciiPrint
`mem_alloc`	gBS->AllocatePool
`mem_free`	gBS->FreePool
`mem_alloc_shared`	gBS->AllocatePool
`mem_free_shared`	gBS->FreePool
`aligned_alloc`	gBS->AllocatePool
`mem_free_aligned`	gBS->FreePool
`mem_get_shared_addr`	None
`mmio_read`	None
`mmio_write`	None

Module-specific APIs

The following table represents the mapping of PAL API to Advanced Configuration and Power Interface (ACPI), if the system firmware presents platform configuration through ACPI tables.

Table 3-3: PAL APIs, UEFI interfaces, and ACPI tables consumed

PAL APIs	UEFI interfaces	ACPI tables consumed
`pe_create_info_table`	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	MADT Table
`call_smc`	-	-
`pe_execute_payload`	-	-
`pe_install_esr`	• gEfiCpuArchProtocolGuid • Cpu->RegisterInterruptHandler	-
`gic_create_info_table`	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	MADT table

PAL APIs	UEFI interfaces	ACPI tables consumed
`gic_install_isr`	• gHardwareInterruptProtocolGuid • RegisterInterruptSource • EnableInterruptSource	-
`timer_create_info_table`	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	GTDT table
`wd_create_info_table`	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	GTDT table
`pcie_create_info_table`	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	MCFG table
`pcie_get_mcfg_ecam`	• gST->ConfigurationTable • CompareGuid, IndustryStandard/Acpi61.h • IndustryStandard/MemoryMappedConfigurationSpaceAccessTable.h	MCFG table
`iovirt_create_info_table`	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	IORT table
`peripheral_create_info_table`	• gEfiPciIoProtocolGuid • Pci->GetLocation • Pci->Pci.Read	-
`memory_create_info_table`	gBS->GetMemoryMap	-

The following table represents the mapping of PAL API to DT node, if the system firmware presents platform configuration through DT nodes.

Table 3-4: PAL APIs, UEFI interfaces, and DT nodes consumed

PAL APIs	UEFI interfaces	DT nodes consumed
`pe_create_info_table`	• gST->ConfigurationTable • CompareGuid	cpu, pmu, interrupt-controller node
`gic_create_info_table`		interrupt-controller, v2m and its nodes
`timer_create_info_table`		systimer and memory mapped timer nodes
`wd_create_info_table`		watchdog nodes
`pcie_create_info_table`		pcie node
`iovirt_create_info_table`		smmu node
`peripheral_create_info_table`		usb, uart, and sata node
`memory_create_info_table`	gBS->GetMemoryMap	-

4. Linux application

This chapter provides information on executing tests from the Linux application.

4.1 Linux application arguments

This section provides information on the Linux application arguments.

Run the Linux application with the following set of arguments.

```
shell> bsa [--v <n>] [--skip <x,y,z>]
```

Table 4-1: Description of Linux application arguments

Argument	Description
`v`	Print level **1** INFO and above **2** DEBUG and above **3** TEST and above **4** WARN and ERROR **5** ERROR
`skip`	Overrides the suite to skip the execution of a particular test. For example, 53 skips test case with ID = 53.

Example

In the following example, the set of parameters tests for compliance against BSA with print verbosity set to 3, and skips the test number 57.

```
shell> bsa --v 3 --skip 57
```

Loading the kernel module

Before running the BSA ACS Linux application, load the BSA ACS kernel module using the `insmod` command.

```
shell> insmod bsa_acs.ko
```

Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

A.1 Revisions

This section consists of all the technical changes between different versions of this document.

Table A-1: Issue 0005-01

Change	Location
First release	-

Table A-2: Differences between Issue 0005-01 to Issue 0009-02

Change	Location
Added the abbreviation for SMMU in the list.	See, 2.1 Abbreviations on page 9.
Added a UEFI shell argument with its description.[-dtb <file name>]	See, 3.1 UEFI shell application arguments on page 11.

Table A-3: Differences between Issue 0009-02 to Issue 0100-01

Change	Location
Added sata to the PAL APIs, UEFI interfaces, and DT nodes consumed table.	See, 3.2 UEFI shell implementation of PAL APIs on page 13.

Table A-4: Differences between Issue 0100-01 to Issue 0100-02

Change	Location
Added abbreviations for HVC and PSCI in the list.	See, 2.1 Abbreviations on page 9.
Added `[-dtb [-t <test id>]`, `[-m <module id>]`, `[-sbsa]`, `[-timeout <wakeup test timeout multiple>]`, `[-p2p]`, `[-cache]` to UEFI shell application arguments	See, 3.1 UEFI shell application arguments on page 11.
Added one more example for UEFI shell application usage.	See, 3.1 UEFI shell application arguments on page 11.

Table A-5: Differences between Issue 0100-02 to Issue 0100-03

Change	Location
Added the parameter bsa.efi to UEFI shell arguments.	See, 3.1 UEFI shell application arguments on page 11.
Added PAL APIs	See, 3.2 UEFI shell implementation of PAL APIs on page 13.

Table A-6: Differences between Issue 0100-03 to Issue 0100-04

Change	Location
Updated the UEFI shell application set of arguments and their descriptions.	See, 3.1 UEFI shell application arguments on page 11.