

Arm[®] SBSA Architecture Compliance Suite

User Guide

Non-Confidential – DEV2.0



Arm®SBSA Architecture Compliance Suite User Guide

Copyright © 2016-2018 Arm Limited or its affiliates. All rights reserved.

Release Information

Document History

Date	Confidentiality	Change
30 November 2016	Non-Confidential	Release for alpha
31 March 2017	Non-Confidential	Release for beta
13 July 2017	Non-Confidential	Release for v1.0
11 May 2018	Non-Confidential	Release for DEV2.0

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.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2016-2018 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

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.

Web Address

<http://www.arm.com>

Contents

Arm® SBSA ACS User Guide

Preface	5
<i>About this book</i>	6
<i>Feedback</i>	10
 Chapter 1	
UEFI Shell application	11
1.1 Overview of tests	12
1.2 UEFI Application arguments	13
1.3 Memory requirements	14
1.4 Interfaces consumed by Shell Application	15
1.5 Toolchain	16
1.6 System dependencies.....	17
1.7 Platform override.....	18
1.8 Test ID	19
1.9 UEFI implementation of PAL APIs	20
 Chapter 2	
Linux kernel module and application	22
2.1 Linux Application arguments.....	23
2.2 SBSA ACS - Linux application	24

Preface

This preface introduces the *Arm® SBSA Architecture Compliance Suite User Guide*.

It contains the following sections:

- [About this book on page 6](#)
- [Feedback on page 10](#)

About this book

This book describes the Architecture Compliance Suite User Guide for the Arm® SBSA architecture.

Product revision status

The *rm**pn* identifier indicates the revision status of the product described in this book, for example, r1p2, where:

rm Identifies the major revision of the product, for example, r1.

pn Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for engineers who are designing or verifying an implementation of the Arm® SBSA architecture.

Using this book

This book is organized into the following chapters:

Chapter 1 UEFI shell application

Read this for information about SBSA Architecture Compliance Suite tests which run as a UEFI Shell Application.

Chapter 2 Linux kernel module and application

Read this for details about SBSA Architecture Compliance Suite tests which run within an Operating System environment.

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.

Typographic conventions

Italic

Introduces special terminology, denotes cross-references, and citations.

bold

Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

monospace

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`

Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`

Denotes language keywords when used outside example code.

<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

Used in body text for a few terms that have specific technical meanings, that are defined in the

Arm Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and

UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

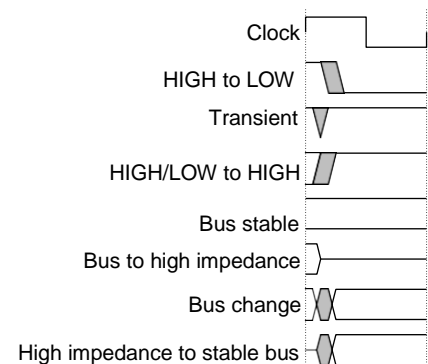


Figure 1 Key to timing diagram conventions**Signals**

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm Publications

- *Server Base System Architecture (ARM-DEN-0029 Version 3.0)*
- *Server Base Boot Requirements (ARM-DEN-0044B)*
- *Arm® Architecture Reference Manual ARMv8, for Armv8-A architecture profile (ARM DDI 0487).*

Other publications

None.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to support-enterprise-accs@arm.com.

Give:

- The title *Arm® SBSA Architecture Compliance Suite User Guide*.
- The number PJDOC-2042731200-3438.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1

UEFI Shell application

This chapter contains the following sections:

- *Overview of tests* on page 12
- *UEFI application arguments* on page 13
- *Memory requirements* on page 14
- *Interfaces consumed by shell application* on page 15
- *Toolchain* on page 16
- *System dependencies* on page 17
- *Platform override* on page 18
- *Test ID* on page 19
- *UEFI implementation of PAL APIs* on page 20

1.1 Overview of tests

The general division of tests between UEFI Shell application and Linux application is as follows.

Table 1-1 Test environment and components covered

Test environment	Components covered
UEFI Shell	PE, GIC, Timers, Watchdog, Wakeup, Secure Devices
Linux command line	PCIe, SMMU

1.2 UEFI application arguments

The application can be run with the following set of arguments:

```
uefi shell> Sbsa.efi [-v <n>] [-l <n>] [-skip <x,y,z>] [-f <file name>] [-s]
```

Table 1-2 Parameters and descriptions

Parameter	Description
v	Print level. 1 – INFO and above. 2 – DEBUG and above. 3 – TEST and above. 4 – WARN and ERROR. 5 – ERROR.
l	Level of compliance to be tested for (0-5).
skip	Overrides the suite to skip the execution of a particular test. <u>Example</u> 33 skips test case with ID 33. 30 skips all tests in module with ID = 30. 50 skips all tests in module with ID = 50. (Refer to test ID section for more details on Module IDs) comma separated. Maximum of three values.
f	File name to which the output log is written.
s	Runs Secure tests before executing Non-secure tests. (Requires Secure firmware code from SBSA ACS to be ported to EL3 FW) Not giving this option runs only Non-secure tests.

For example,

```
Shell > sbsa.efi -v 2 -l 3 -f acs.txt -skip 20,36
```

This set of parameters:

- Prints messages with verbosity of 2 and above.
- Tests for compliance against SBSA level 3.
- Skips execution of all tests belonging to GIC module and test number 36.
- Stores the log messages to file `acs.txt`.

1.3 Memory requirements

Code

Binary size – 165KB

Data

EfiBootServicesData

Table 1-3 Data Structure and Size

Data Structure	Size (in Bytes)
PE_INFO_TABLE	8192
GIC_INFO_TABLE	2048
TIMER_INFO_TABLE	1024
WD_INFO_TABLE	512
PCIE_INFO_TABLE	64
PERIPHERAL_INFO_TABLE	1024
IO-Virtualization Table	2048
PE_SHARED_MEMORY	(num_of_pe) * 16
PE_SECONDARY_STACK	(num_of_pe) * 256
Total (Assuming 48 PEs)	27,992

EfiRuntimeServicesData

None

1.4 Interfaces consumed by shell application

Libraries

- UefiLib
- ShellLib
- BaseMemoryLib
- ShellCEntryLib
- UefiBootServicesTableLib
- UefiRuntimeServicesTableLib

Protocols

- gEfiAcpiTableProtocolGuid
- gHardwareInterruptProtocolGuid
- gEfiPciIoProtocolGuid

1.5 Toolchain

Linaro AArch64 5.3 toolchain was used to develop this application.

The toolchain is at:

<http://releases.linaro.org/components/toolchain/binaries/5.3-2016.02/aarch64-linux-gnu/>

1.6 System dependencies

PSCI

The compliance suite makes the following PSCI calls:

ARM_SMC_ID_PSCI_CPU_SUSPEND_AARCH64	(0xc4000001)
ARM_SMC_ID_PSCI_CPU_OFF	(0x84000002)
ARM_SMC_ID_PSCI_CPU_ON_AARCH64	(0xc4000003)

1.7 Platform override

On certain platforms, the underlying ACPI infrastructure to provide information on the system is not implemented yet. To enable running SBSA ACS on these platforms, override hooks are provided for certain modules which take the relevant hardware information from the override file rather than the underlying UEFI framework.

See `<acs_local_path>/sbsa-acs/platform/pal_uefi/include/platform_override.h` file in the source code for available options.

1.8 Test ID

Test ID of each test is generated as an addition of Module ID and Unit Test ID.

For a given module, Unit Test ID begins from 1.

Module IDs are as follows.

Table 1-4 Module Name and Module ID

Module Name	Module ID
PE	0
GIC	20
Timer	30
Watchdog	40
PCIe	50
Power & Wakeup	70
Peripheral	80
SMMU	90
Secure	900

1.9 UEFI implementation of PAL APIs

The following table lists the UEFI interfaces used for the implementation of the *Platform Abstraction Layer* (PAL) APIs mentioned in the *SBSA Validation Methodology Document*. See https://github.com/ARM-software/sbsa-acs/tree/master/docs/SBSA_Val_Methodology.pdf

1.9.1 Infrastructure APIs

Table 1-5 PAL APIs and UEFI interfaces

PAL API	UEFI interfaces
Pal_print	AsciiPrint
Mem_alloc	gBS->AllocatePool
Mem_free	gBS->FreePool
Mem_alloc_shared	gBS->AllocatePool
Mem_free_shared	gBS->FreePool
Mem_get_shared_addr	None
Mmio_read	None
Mmio_write	None

Module-specific APIs

Table 1-6 PAL APIs, UEFI interfaces, and ACPI tables consumed

PAL API	UEFI interfaces consumed	ACPI table consumed
Pe_create_info_table	<ul style="list-style-type: none"> gST->ConfigurationTable CompareGuid IndustryStandard/Acpi61.h 	MADT Table
Call_smc	None	-
Pe_execute_payload	None	-
Pe_install_esr	<ul style="list-style-type: none"> gEfiCpuArchProtocolGuid Cpu->RegisterInterruptHandler 	-
Gic_create_info_table	<ul style="list-style-type: none"> gST->ConfigurationTable CompareGuid IndustryStandard/Acpi61.h 	MADT table
Gic_install_isr	<ul style="list-style-type: none"> gHardwareInterruptProtocolGuid RegisterInterruptSource EnableInterruptSource 	-
Timer_create_info_table	<ul style="list-style-type: none"> gST->ConfigurationTable CompareGuid IndustryStandard/Acpi61.h 	GTDT Table
Wd_create_info_table	<ul style="list-style-type: none"> gST->ConfigurationTable CompareGuid IndustryStandard/Acpi61.h 	GTDT Table
Pcie_create_info_table	<ul style="list-style-type: none"> gST->ConfigurationTable CompareGuid IndustryStandard/Acpi61.h 	MCFG Table
Pcie_get_mcfg_ecam	<ul style="list-style-type: none"> gST->ConfigurationTable CompareGuid, IndustryStandard/Acpi61.h 	MCFG Table

PAL API	UEFI interfaces consumed	ACPI table consumed
	<ul style="list-style-type: none"> IndustryStandard/MemoryMappedConfigurationSpaceAccessTable.h 	
Iovirt_create_info_table	<ul style="list-style-type: none"> gST->ConfigurationTable CompareGuid IndustryStandard/Acpi61.h 	IORT Table
Peripheral_create_info_table	<ul style="list-style-type: none"> gEfiPciIoProtocolGuid Pci->GetLocation Pci->Pci.Read 	-
Memory_create_info_table	gBS->GetMemoryMap	-

Chapter 2

Linux kernel module and application

This chapter contains the following sections:

- [Linux application argument on page 23](#)
- [SBSA ACS - Linux application on page 24](#)

2.1 Linux application arguments

The application can be run with the following set of arguments:

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

Table 2-1 Parameter and its description

Parameter	Description
v	Print Level. 1 – INFO and above 2 - DEBUG and above 3 – TEST and above 4 – WARN and ERROR 5 - ERROR
l	Level of compliance to be tested for. (0 to 5)
skip	Overrides the suite to skip the execution of a particular test. <u>Example</u> 53 skips test case with ID 53.

For example,

```
shell> sbsa --v 3 --l 3 --skip 57
```

This set of parameters tests for compliance against SBSA level 3 with print verbosity set to 3 and skips test number 57.

2.1.1 Loading the kernel module

Before the SBSA ACS Linux application can be run, the SBSA ACS kernel module must be loaded. This can be achieved by using the `insmod` command.

For example,

```
Shell>insmod sbsa_acs.ko
```

2.2 SBSA ACS - Linux application

2.2.1 Source path

The patch for the kernel tree and the Linux Platform Abstraction Layer are hosted separately on <http://linux-arm.org/git?p=linux-accs.git;a=summary>.

2.2.2 Kernel module – build

Prerequisites

- Linux kernel source version 4.14.
- Linaro GCC tool chain 5.3 or above.
- Build environment for AArch64 Linux kernel.

Porting steps – Linux kernel

1. `git clone git://linux-arm.org/linux-accs.git <local_dir/sbsa-accs-drv>`
2. `git clone https://github.com/ARM-software/sbsa-accs.git <local_dir/sbsa-accs>`
3. Apply the `<local_dir>/kernel/src/0001-Enterprise-accs-linux-v4.13.patch` patch to your kernel source tree.
4. Build the kernel.

Build steps – SBSA kernel module

1. `cd <local_dir>/sbsa-accs-drv/files`
2. Set `CROSS_COMPILE` to the ARM64 toolchain path.
3. `export KERNEL_SRC=<linux kernel path>`
4. `./setup.sh <local_dir/sbsa-accs>`
5. `./linux_sbsa_acs.sh`
`sbsa_acs.ko` file is generated.

2.2.3 SBSA Linux application build

1. `cd <sbsa-accs path>/linux_app/sbsa-accs-app`
2. Set `CROSS_COMPILE` to the ARM64 toolchain path.
`export CROSS_COMPILE=<local_dir>/gcc-linaro-5.3-2016.02/bin/aarch64-linux-gnu-`
3. `make`

Executable file: `sbsa` is generated.

2.2.4 Target environment setup

The set of tests assumes at least one SATA controller behind a PCIe root complex. The SATA controller may or may not be behind an IOMMU.

Before running these tests, at least one SATA hard disk must be connected to the SATA controller.

The test performs read and write operations to the SATA hard disk. Therefore, the data on the HDD is overwritten. The SATA drive must not be the boot device for the OS.

2.2.5 Runtime environment

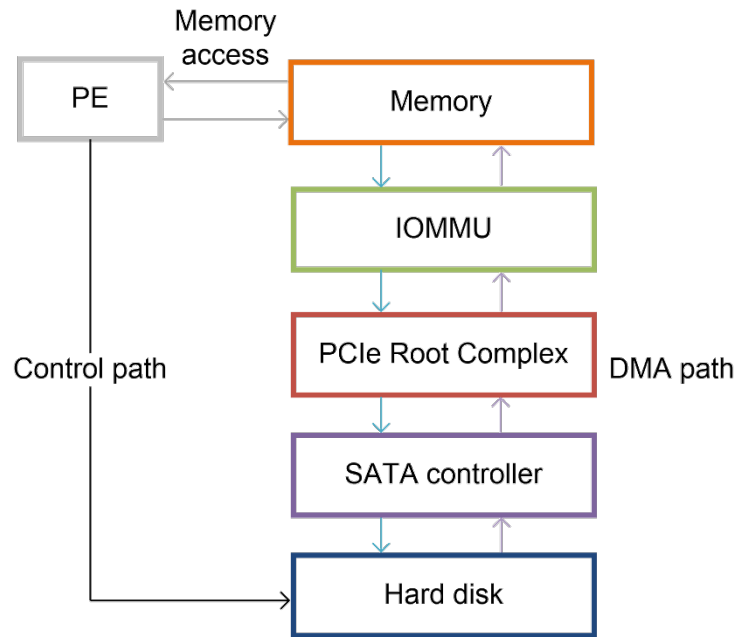


Figure 2-1 Hardware functional blocks

The PCIe-DMA tests initiate data transfers from a DMA master. By default, the test searches for a SATA controller which is part of the PCIe subsystem.

1. The test writes known data from the PE to main memory.
2. The test programs the DMA master to transfer this known data to its end-point device.
3. The test asks the DMA master to transfer the data back to a different location in the main memory.
4. The test compares the data at both the locations.

Also, if the SATA controller is not behind an IOMMU, during this data transfer, the address that is used by the SATA controller is retrieved and compared with the DMA address that is seen by the PE.

If the DMA master is behind an IOMMU, then the address that is used by the SATA AHCI controller is compared with the address that is seen by the IOMMU. Both these addresses must match.

To enable the export of the addresses that are seen by the SATA AHCI controller and IOMMU, the kernel drivers for these two modules must be patched.