ARM®SBSA Architecture Compliance Suite

Revision: r1p0

Application User Guide

Non-confidential

ARM

ARM-EPM-123720 v1.0

ARM®SBSA Architecture Compliance Suite
Application User Guide

Release Information
The following changes have been made to this document.

Change History

Issue	Date	Confidentiality	Change
A	30 November 2016	Non-confidential	Release for alpha
B	31 March 2017	Non-confidential	Release for beta
C	13 July 2017	Non-confidential	Release for v1.0

Confidentiality Status
This document is non-confidential.

Product Status
The information in this document is for a REL v1.0 product.

Web Address
http://www.arm.com

Contents

ARM®SBSA ACS Application User Guide

ARM®SBSA Architecture Compliance Suite 6
Application User Guide 6

Preface ***6***
About this book 7
Using this book 8
Conventions 9
Typographic conventions 9
Numbers 9
Additional reading 9
ARM Publications 10
Feedback 10

UEFI Shell application ***11***
1.1 Overview of tests 12
1.2 UEFI Application arguments 12
1.3 Memory requirements 13
Code 13
Data 13
1.4 Interfaces consumed by Shell Application 13
Libraries 13
Protocols 13
1.5 Toolchain 14
1.6 System dependencies 14
1.6.1 PSCI 14
1.7 Platform override 14
1.8 Test ID 14
1.9 UEFI implementation of PAL APIs 15
1.9.1 Infrastructure APIs 15
1.9.2 Module-specific APIs 15

Chapter 2 ..... ***17***

Linux kernel module and application ***17***
2.1 Linux Application arguments 18
2.1.1 Loading the kernel module 18
2.2 SBSA ACS - Linux application 18
2.2.1 Source path 18
2.2.2 Kernel module – build 18
2.2.3 SBSA Linux application build 19
2.2.4 Target environment setup 19
2.2.5 Runtime environment 19

Preface

This preface introduces the *ARM® SBSA Architecture Compliance Suite User Guide*. It contains the following sections:

- *About this book* on page x.
- *Using this book* on page xi.
- *Conventions* on page xii.
- *Additional reading* on page xiii.
- *Feedback* on page xiv.

About this book

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

Intended audience

This book is written for engineers who are specifying, 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.

Conventions

The following sections describe conventions that this book can use:

- *Typographic conventions*.
- *Numbers*.

Typographic conventions

The typographical conventions are:

italic	Introduces special terminology, and denotes citations.
bold	Denotes signal names, and is used for terms in descriptive lists, where appropriate.
monospace	Used for assembler syntax descriptions, pseudocode, and source code examples. Also used in the main text for instruction mnemonics and for references to other items appearing in assembler syntax descriptions, pseudocode, and source code examples.
SMALL CAPITALS	Used for a few terms that have specific technical meanings.
Colored text	Indicates a link. This can be: • A URL, for example http://infocenter.arm.com. • A cross-reference, that includes the page number of the referenced information if it is not on the current page, for example, *Feedback on this product* on page xi. • A link, to a chapter or appendix, or to the section of the document that defines the colored term, for example Appendix A *Memory Attributes*.

Numbers

Numbers are normally written in decimal. Binary numbers are preceded by 0b, and hexadecimal numbers by 0x. In both cases, the prefix and the associated value are written in a monospace font, for example 0xFFFF0000.

Additional reading

This document refers to the following external documents.

- http://www.uefi.org/specifications
  - ACPI Specification Version 6.1
  - UEFI Specification Version 2.6
  - UEFI Platform Initialization Specification Version 1.4 (Volume 2)
- http://www.uefi.org/acpi
  - WDRT
  - MCFG
  - IORT
  - DBG2
- For general introduction and build steps for SBSA ACS, see https://github.com/ARM-software/sbsa-acs/blob/master/README.md
- For more details on the Validation Methodology, see https://github.com/ARM-software/sbsa-acs/tree/master/docs.

See Infocenter: http://infocenter.arm.com for access to ARM documentation

ARM Publications

This document refers to the following documents:

- *Server Base System Architecture (ARM-DEN-0029 Version 3.0)*
- *Server Base Boot Requirements (ARM-DEN-0044B Version 1.0)*
- *ARM® Architecture Reference Manual ARMv8, for ARMv8-A architecture profile (ARM DDI 0487).*

Feedback

ARM welcomes feedback on its documentation.

Chapter 1

UEFI Shell application

This chapter contains the following sections:

- *Overview of tests on page 16*
- *Application arguments on page 16*
- *Memory requirements on page 17*
- *Interfaces consumed by Shell Application on page 17*
- *Toolchain on page 18*
- *System Dependencies on page 18*
- *Platform Override on page 18*
- *Test ID on page 18*
- *UEFI implementation of PAL APIs on page 18*

1.1 Overview of tests

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

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 1-1 Parameter and its description

Parameter	Description
v	Print Level. 1 – INFO & above. 2 – DEBUG & above. 3 – TEST & above. 4 – WARN & ERROR. 5 – ERROR.
l	Level of compliance to be tested for (0-3).
skip	Overrides the suite to skip the execution of a particular test. Example 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 will:

- Print messages with verbosity of 2 and above.
- Test for compliance against SBSA level 3.
- Skip execution of all tests belonging to GIC module and test number 36.
- Store the log messages to file `acs.txt`.

1.3 Memory requirements

Code

Binary size – 165KB

Data

EfiBootServicesData

Table 1-2 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 located at:

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

1.6 System dependencies

1.6.1 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 1-3 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.* **(**https://github.com/ARM-software/sbsa-acs/tree/master/docs/SBSA_Val_Methodolgy.pdf**)**

1.9.1 Infrastructure APIs

Table 1-4 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

1.9.2 Module-specific APIs

Table 1-5 PAL APIs, UEFI interfaces, and ACPI tables consumed

PAL API	UEFI interfaces consumed	ACPI table consumed
Pe_create_info_table	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	MADT Table
Call_smc	None	-
Pe_execute_payload	None	-
Pe_install_esr	• gEfiCpuArchProtocolGuid • Cpu->RegisterInterruptHandler	-
Gic_create_info_table	• gST->ConfigurationTable • CompareGuid • IndustryStandard/Acpi61.h	MADT table
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	MCFG Table

PAL API	UEFI interfaces consumed	ACPI table consumed
	• CompareGuid, IndustryStandard/Acpi61.h • IndustryStandard/MemoryMappedConfigurationSpaceAccessTable.h	
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	-

Chapter 2
Linux kernel module and application

This chapter contains the following sections:

- *Application Arguments on page 22*
- *SBSA ACS - Linux Application on page 22*

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 & above 2 - DEBUG & above 3 – TEST & above 4 – WARN & ERROR 5 - ERROR
l	Level of compliance to be tested for. (0 to 3)
skip	Overrides the suite to skip the execution of a particular test. Example 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-acs.git;a=summary.

2.2.2 Kernel module – build

Pre-requisites

1. Linux kernel source version 4.10.
2. Linaro GCC tool chain 5.3 or above.
3. Build environment for AArch64 Linux kernel.

Porting steps – Linux kernel

1. `git clone git://linux-arm.org/linux-acs.git <local_dir/sbsa-acs-drv>`
2. `git clone https://github.com/ARM-software/sbsa-acs.git <local_dir/sbsa-acs>`
3. Apply the `<local_dir>/kernel/src/001-SBSA-ACS-linux-4.10` patch to your kernel source tree.
4. Build the kernel.

Build steps – SBSA kernel module

1. `cd <local_dir>/sbsa-acs-drv/files`
2. Set the GCC path to the ARM64 toolchain.
3. `export KERNEL_SRC=<linux kernel path>`
4. `./setup.sh <local_dir/sbsa-acs>`
5. make

`sbsa_acs.ko` file is generated.

2.2.3 SBSA Linux application build

1. `cd <sbsa-acs path>/linux_app/sbsa-acs-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 might or might 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. Hence, the data on the HDD is overwritten. The SATA drive must not be the boot device for the OS.

2.2.5 Runtime environment

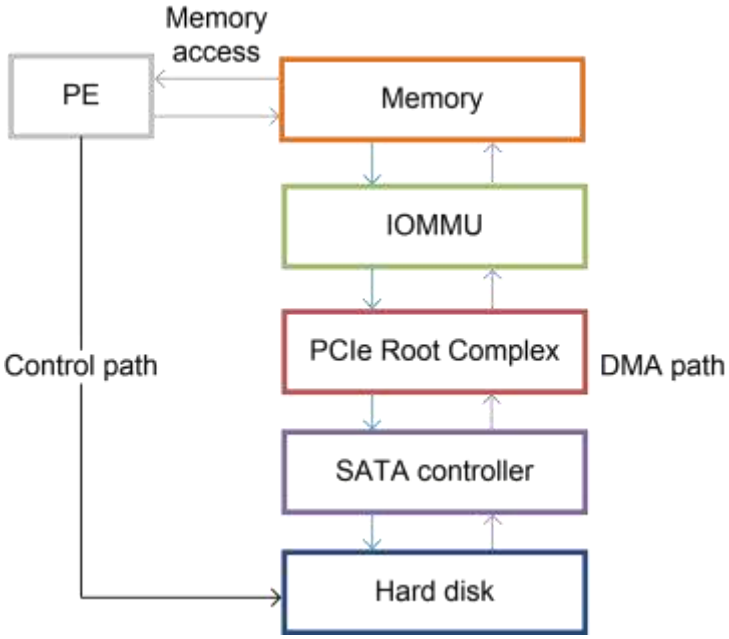

Figure 2-1 Hardware functional blocks

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.