ARM®SBSA Architecture Compliance Kit

Revision: r0p1

Application User Guide

Non-Confidential – Beta

ARM® SBSA Architecture Compliance Kit
UEFI Shell 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

Proprietary Notice

Confidentiality Status
This document is non-confidential.

Product Status
The information in this document is for a Beta product, that is a product under development.

Web Address
http://www.arm.com

Contents

ARM®SBSA UEFI Shell Application User Guide

Document Overview

This document is divided into 2 sections.

The first half describes the SBSA Architecture Compliance Suite tests which run as a UEFI Shell Application.

The second half describes the SBSA ACS tests which run within an Operating system environment.

SBSA ACS Overview

For general introduction and Build steps for SBSA ACS, please refer to https://github.com/ARM-software/sbsa-acs/blob/master/README.md

For more details on the Validation Methodology, please refer to **https://github.com/ARM-software/sbsa-acs/tree/master/docs.**

The general division of Tests between UEFI Shell Application and Linux Application are as follows.

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

PART 1 – UEFI Shell Application

1.1 Application arguments

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

Parameters	Description
V	This is for the Print Level. 1 – DEBUG & above 2 – INFO & above 3 – TEST & above 4 – WARN & ERROR 5 - ERROR
l	This is for the level of compliance to be tested against. (0 thru
skip	This will override the suite to skip the execution of a particular test(s). <u>Example</u> 33 will skip test case with ID 33. 30 will skip all tests in module with ID = 30. 50 will skip all tests in module with ID = 50. (refer to test ID section below for more details on Module IDs) comma separated. Maximum of 3 values.
f	File name to which the output log is written
s	This will run secure tests before executing non-secure tests. (requires secure firmware code from SBSA ACS to be ported to EL3 FW) Not giving this option will run only non-secure tests

Example

```
Shell>Sbsa.efi –v 2 –l 3 –f acs.txt –skip 10,36
```

These 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

1.2 Memory requirements

Code

Binary size – 153KB

Data

EfiBootServicesData

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

1.3 Interfaces consumed by Shell Application

Libraries

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

Protocols

- gEfiAcpiTableProtocolGuid
- gHardwareInterruptProtocolGuid
- gEfiPciIoProtocolGuid

1.4 Toolchain

Linaro aarch64 5.3 toolchain was used to compile this application.

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

1.5 System Dependencies

1.5.1 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)

1.6 Platform Override

It is anticipated that 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 will 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.

1.7 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.

Module Name	**Module ID**
PE	0
GIC	20
Timer	30
Watchdog	40
PCIe	50
SMMU	60
Power & Wakeup	70
Peripheral	80
Secure	900

1.8 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)**

1.8.1 Infrastructure APIs

PAL API	UEFI Interfaces used
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

1.8.2 Module Specific APIs

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
Timer_system_start_count down	To be implemented	
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	

PART – 2 Linux Kernel Module and Application

2.1 Application arguments

shell> sbsa [-l <n>]

Parameters	Description
L	This is for the level of compliance to be tested against. (0 thru

Example

Shell>Sbsa –l 3

These set of parameters will test for compliance against SBSA level 3.

2.1.1 Loading the Kernel Module

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

Example –

Shell>insmod sbsa_acs.ko

2.2 SBSA ACS - Linux Application

2.2.1 Kernel Module - Build

Pre-requisites

1. Linux Kernel source version 4.9 or above.
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>`**
2. Apply the <local_dir>/kernel/src/001-SBSA-ACS-linux-4.10.patch to your Kernel source tree.
3. 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. Make

sbsa_acs.ko file is generated.

2.2.2 SBSA Linux Application Build

1. cd <sbsa-acs path>/linux_app/sbsa-acs-app
2. set GCC49_AARCH64_PREFIX to the ARM64 toolchain path.
    a. export GCC49_AARCH64_PREFIX=<local_dir>/gcc-linaro-5.3-2016.02/bin/aarch64-linux-gnu-
3. make

Executable file: "sbsa" is generated.

2.2.3 Target Environment – Setup

The present set of tests makes the assumption of at least one SATA controller behind a PCIe root complex.

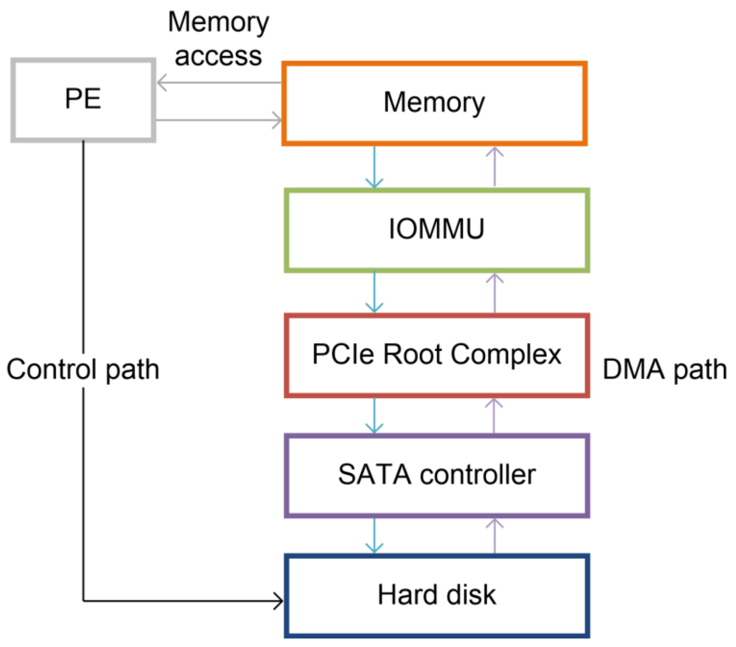
The SATA controller may or may not be behind an IOMMU.

Before running these tests, it is required that at least one SATA hard disk is connected to the SATA controller.

The test will perform read and write operations to the SATA hard disk. Hence, the Data on the HDD will be over-written. It is imperative that the SATA drive was not the boot device for the operating system.

2.2.4 Runtime Environment

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 sub-system.

1. The test first writes known data from the PE to Main memory.
2. The test then 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 used by the SATA controller is retrieved and compared with the DMA address seen by the PE.

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

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

The patch for the kernel tree is hosted separately on the http://linux-arm.org/git?p=linux-acs.git;a=summary.