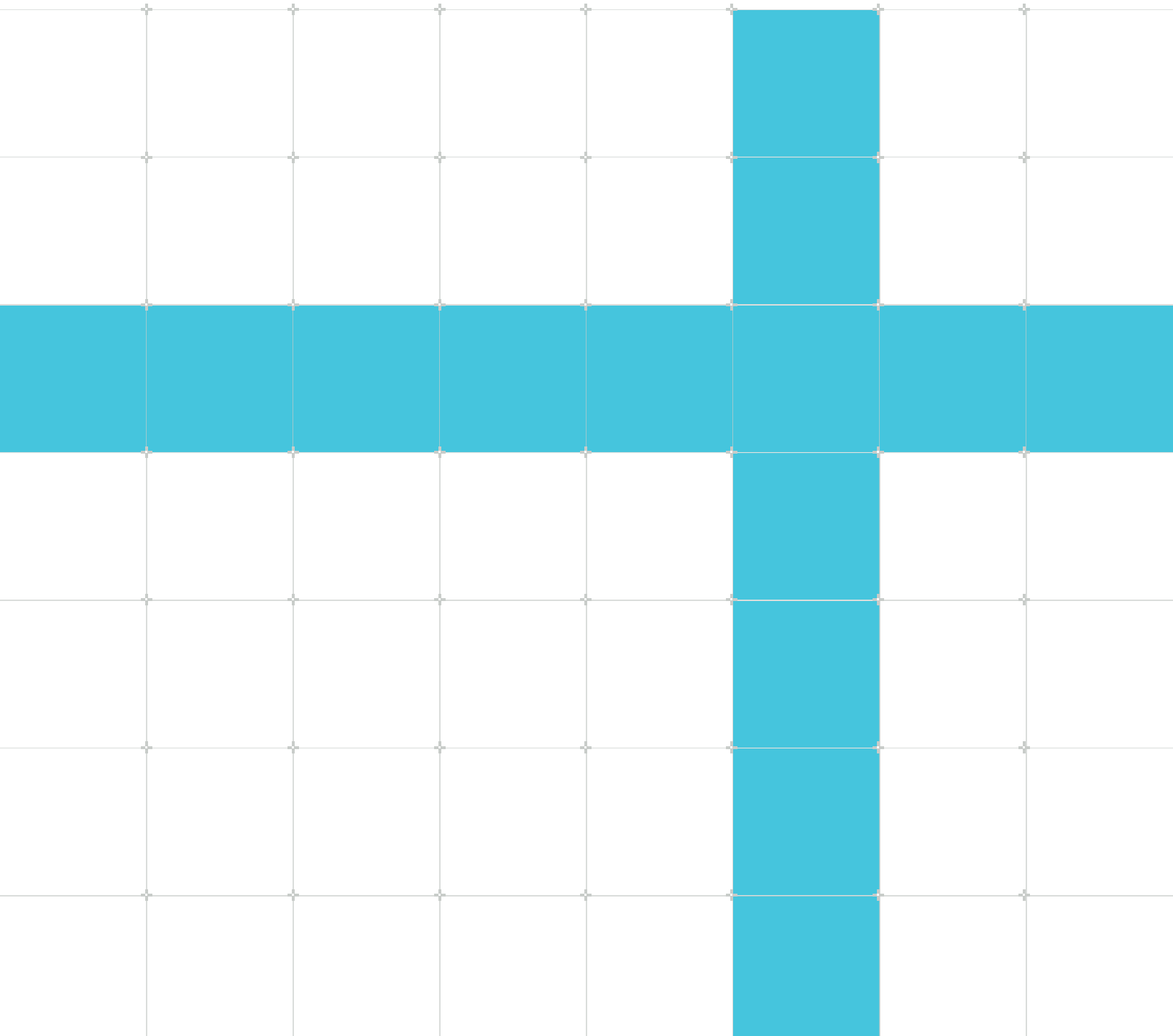
Arm® Lumex™ Reference Software

Version 24.3

User Guide

Non-Confidential

Issue 04
109541_24.3_04_en



Arm® Lumex™ Reference Software User Guide

This document (109541_24.3_04_en) was issued on 2025-09-24. There might be a later issue at https://developer.arm.com/documentation/109541

The product version is 24.3.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This document is intended for Arm partners creating System on Chip (SoC) designs. It will be of interest to software developers using the Lumex software stack and associated Fixed Virtual Platform.

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.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

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.

Contents

1. Overview of Lumex Reference Software

The Arm Lumex™ Reference Software (Lumex) is a fully integrated open-source software stack, from firmware up to Android. You can use the software stack, along with a Fixed Virtual Platform (FVP) to modify, extend, and develop the software for a *System on Chip* (SoC) based on Lumex. The software stack and FVP enable early testing, system integration, and validation, reducing time to market and boosting product security and reliability.

Deliverables

Arm provides documentation, an FVP, and a reference software stack to help you develop software.

Lumex Reference Software User Guide

This guide provides a high-level overview of Lumex CSS, the associated software stack, and the FVP.

Fixed Virtual Platform

The Fixed Virtual Platform (FVP) provides a software model of the Lumex reference mobile configuration design. The FVP models the programmer's view of the design, allowing programmers to execute software without an actual hardware platform. The execution speeds make it possible to run Linux and Android on the model. The Lumex software stack executes directly on the FVP.

Lumex Reference Software Stack

The software stack that covers all the necessary software components for a client platform, from low-level firmware to the high-level OS, including:

- The Secure software components that run in the Secure world of the design
- A starting point to modify, extend, and develop software for a SoC similar to Lumex
- The FVP is used with the Lumex reference software stack. For instructions on software, and how to set up and run the FVP, see Lumex on the Arm Developer website.

Features of Lumex CSS Platform

Lumex CSS Platform provides the following key features:

- Support for Arm® v9.3-A architecture with a combination of three tiers of processor cores, optimized for both power efficiency and performance.
- Support for *Scalable Matrix Extension* version 2 (SME2) through *Compute Matrix Engines* (CMEs), which accelerates matrix operations for machine learning and computer vision workloads while consuming less power.
- Arm® Mali™ G1-Ultra core with 5th generation architecture for running advanced graphics and computing APIs, which significantly benefits General Purpose computing on GPU (GPGPU) applications.
- Dedicated Neural Engines inside each GPU shader core for boosting machine learning performance.
- Arm® SI L1 System Interconnect *Network-on-Chip* (NoC) provides a flexible interconnect to link various components inside the subsystem and provide expansion interfaces for SoC-level components.

- SI L1 *Memory Controller Nodes* (MCNs) significantly reduces the memory access latency between processor cores and the DRAM compared to the previous generation interconnect. It further reduces the memory access latency and power consumption with built-in *System Level Cache* (SLC).
- A *System Control Processor* (SCP) Block for low-level system management, such as power, thermal, and sensors.
- A *Runtime Security Engine* (RSE) Block brings a hardware-enhanced Secure Enclave into the subsystem. It is the *Root of Trust* (RoT) and provides security services to the rest of the subsystem.

1.1 Lumex CSS overview

The Lumex CSS is a configurable combination of selected IP components that can be used in a mobile device compute subsystem.

Lumex CSS contains the following CPUs, GPU, and System IP:

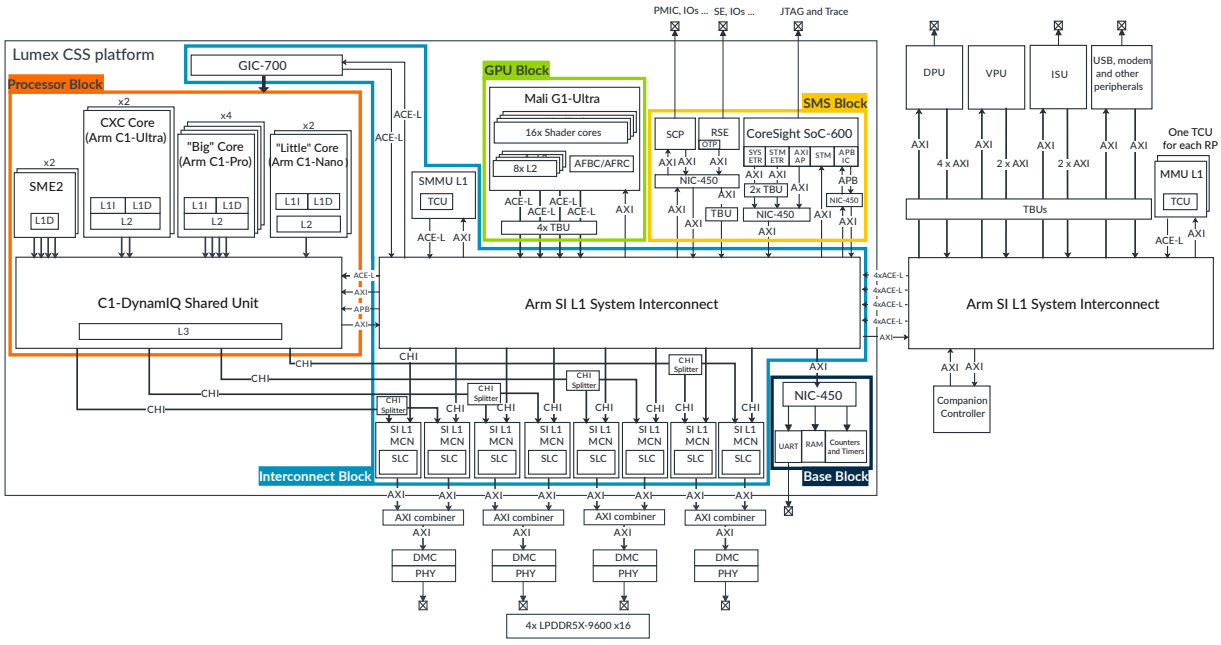
- Armv9.3-A processor cores and *Compute Matrix Engines* (CME) with *DynamIQ™ Shared Unit* (DSU) connect
- Arm® C1-Ultra Core
- Arm® C1-Pro Core
- Arm® C1-Nano Core
- SME2 Compute Matrix Engine
- C1-DynamIQ™ Shared Unit
- Mali™ G1-Ultra Core
- SI L1 System Interconnect
- Cortex®-M55-based *Runtime Security Engine* (RSE)
- Cortex®-M85-based *System Control Processor* (SCP)
- CoreSight™ SoC-600-based debug and profiling

1.2 System architecture

The Lumex CSS Platform architecture is partitioned into functional design blocks that combine major IP and their supporting logic. Some features of the design incorporate functionality from multiple blocks. This block-based design approach provides flexibility, scalability, and modularity.

The following figure provides a high-level architectural view of Lumex CSS Platform.

Figure 1-1: System architecture of Lumex



Note

The *Fixed Virtual Platform* (FVP) models a subset of the Lumex components. For more information, see Introduction to the Lumex FVP.

2. Introduction to the Lumex FVP

Arm *Fixed Virtual Platform* (FVP) models use Arm Fast Models technology to deliver fast simulations of Arm-based systems. They allow you to develop software ahead of hardware availability and to explore the design from a software perspective. You can efficiently develop software and firmware by using the FVP together with the corresponding software stack, reducing the amount of development work.

2.1 About the Lumex FVP

The Arm FVP models a *Programmer's View* (PV) of processors and other devices in a system. A PV provides functional behavior equivalent to what a programmer sees using the hardware. The PV sacrifices timing accuracy to achieve fast simulation execution speeds. Therefore, you can use the FVP to confirm software functionality, but must not rely on the accuracy of cycle counts, low-level component interactions, or other hardware-specific behavior. Neither can you use the FVP to measure software performance.


Note

Lumex FVPs are standalone executables for Linux. They are not customizable, although you can configure some aspects of their behavior with command-line parameters as described in Launch the FVP.

For more information on the FVPs, see the following documentation:

- *Fast Models Fixed Virtual Platforms (FVP) Reference Guide*
- *Fast Models Reference Guide*

See Introduction to the Lumex software stack for more information about the software stack components.

The Lumex *Fixed Virtual Platform* (FVP) models a premium mobile subsystem.

The FVP models the following key components inside the subsystem:

- Arm® C1-Ultra Core x 2
- Arm® C1-Pro Core x 4
- Arm® C1-Nano Core x 2
- Arm® C1-DynamIQ™ Shared Unit
- Arm® SME2 Compute Matrix Engine
- Arm® Mali™ G1-Ultra core
- Arm® Mali™-D71 Display Processor
- Arm® SI L1 System Interconnect

- Arm® CoreLink™ GIC-700 Generic Interrupt Controller
- Arm® MMU L1 System Memory Management Unit
- *System Control Processor* (SCP) based on the Cortex®-M85
- Arm® Runtime Security Engine (RSE-100) based on the Cortex®-M55
- Arm® CoreLink™ NIC-450 Network Interconnect
- Arm® CoreLink™ TZC-400 Address Space Controller
- Arm® CoreLink™ PCK-600 Power Control Kit
- On-Chip ROM, RAM, and other peripherals
- Clock generators with support for dynamic clock gating

To provide a complete system boot environment, the FVP models components that reside outside the compute subsystem, including DRAM memory and external peripherals. Together, these components are referred to as the Rest of System.

The FVP does not model the following components:

- Arm® CoreSight™ System-on-Chip SoC-600
- *Dynamic Memory Controller* (DMC)

2.2 Rest of System

The *Rest of the System* (RoS) contains the components that do not reside in the Lumex. These components create the necessary environment for running and developing a software stack.

The Lumex *Fixed Virtual Platform* (FVP) includes and represents three main parts of the system:

- *Compute Subsystem* (CSS)
- *System on Chip* (SoC) that contains the CSS
- Board that contains the SoC

The last two of these three parts are covered by the RoS.

Similar to how the CSS integrates into an SoC and a board, the RoS components connect to the CSS through its expansion interfaces. This adds to the expansion sections of the memory map and interrupt map of the CSS.

2.2.1 FVP RoS memory map and interrupts

The following table lists the RoS IP, its memory map, and interrupts for the FVP platform.


Note

- The base address for the Lumex RoS memory map is `0x6000_0000`. The table lists the Start and End address offsets that extend from the base address. For example, to access the Virtio Block Device, use `0x6002_0000-0x6002_FFFF`.
- The base interrupt ID for the AP core is `800`. The table lists the interrupt ID offset. For example, for the RTC use interrupt `809`.
- From the *SoC Management Subsystem* (SMS), you must program through the SCP *Address Translation Unit* (ATU) in the range `0xF000_0000-0xFFFF_FFFF`. The ATU translates this to addresses in the range `0x6000_0000-0x6FFF_FFFF`.

Table 2-1: FVP RoS memory map and interrupts

Name	Start	End	Size	Interrupt	Comments
Reserved	10000	1FFFF	64 KB	0	-
Virtio Block Device	20000	2FFFF	64 KB	1	Virtio block device
Virtio RNG	30000	3FFFF	64 KB	2	Random Number Generator component to generate entropy
Virtio Net	40000	4FFFF	64 KB	3	Virtio Net device over *Memory-Mapped I/O* (MMIO) transport
Virtio_P9	50000	5FFFF	64 KB	4	Virtio P9 server
VSI 0	60000	6FFFF	64 KB	5	Arm VHT virtual stream interface
VSI 1	70000	7FFFF	64 KB	6	Arm VHT virtual stream interface
Juno_sysreg	80000	8FFFF	64 KB	-	Registers only - no interrupt request (IRQ) connected
Virtio Net (2)	90000	9FFFF	64 KB	8	Extra Virtio Net
RTC	A0000	AFFFF	64 KB	9	PL031 real time clock
PL180_MCI	B0000	BFFFF	64 KB	10, 11	Arm® PrimeCell Multimedia Card Interface (PL180) - 2 interrupts
Reserved	C0000	CFFFF	64 KB	12	-
Reserved	D0000	DFFFF	64 KB	13	-
Reserved	E0000	EFFFF	64 KB	14	-
Reserved	F0000	FFFFF	64 KB	15	-
PL050_KMI 0	100000	10FFFF	64 KB	16	Arm® PrimeCell PS2 Keyboard/Mouse Interface (PL050)
PL050_KMI 1	110000	11FFFF	64 KB	17	Arm® PrimeCell PS2 Keyboard/Mouse Interface (PL050)
Reserved	120000	12FFFF	64 KB	18	-
Reserved	130000	13FFFF	64 KB	19	-
Reserved	140000	14FFFF	64 KB	20	-
PL011_UART 0	150000	150FFF	4 KB	21	Arm® PL011 PrimeCell UART
PL011_UART 1	151000	151FFF	4 KB	22	Arm® PL011 PrimeCell UART
PL011_UART 2	152000	152FFF	4 KB	23	Arm® PL011 PrimeCell UART
PL011_UART 3	153000	153FFF	4 KB	24	Arm® PL011 PrimeCell UART
RNG	154000	154FFF	4 KB	25	Random Number Generator
Reserved	155000	155FFF	4 KB	26	-

Name	Start	End	Size	Interrupt	Comments
Reserved	156000	156FFF	4 KB	27	-
Reserved	157000	157FFF	4 KB	28	-
Reserved	158000	158FFF	4 KB	29	-
Reserved	-	-	-	30	-
CS2 - SMSC_91C111	4000000	7FFFFFF	64 MB	31	Ethernet
Flash	8000000	FFFFFFFF	128 MB	-	Flash memory

2.2.2 DPU memory map

The following table lists the DPU memory map.

Table 2-2: DPU memory map

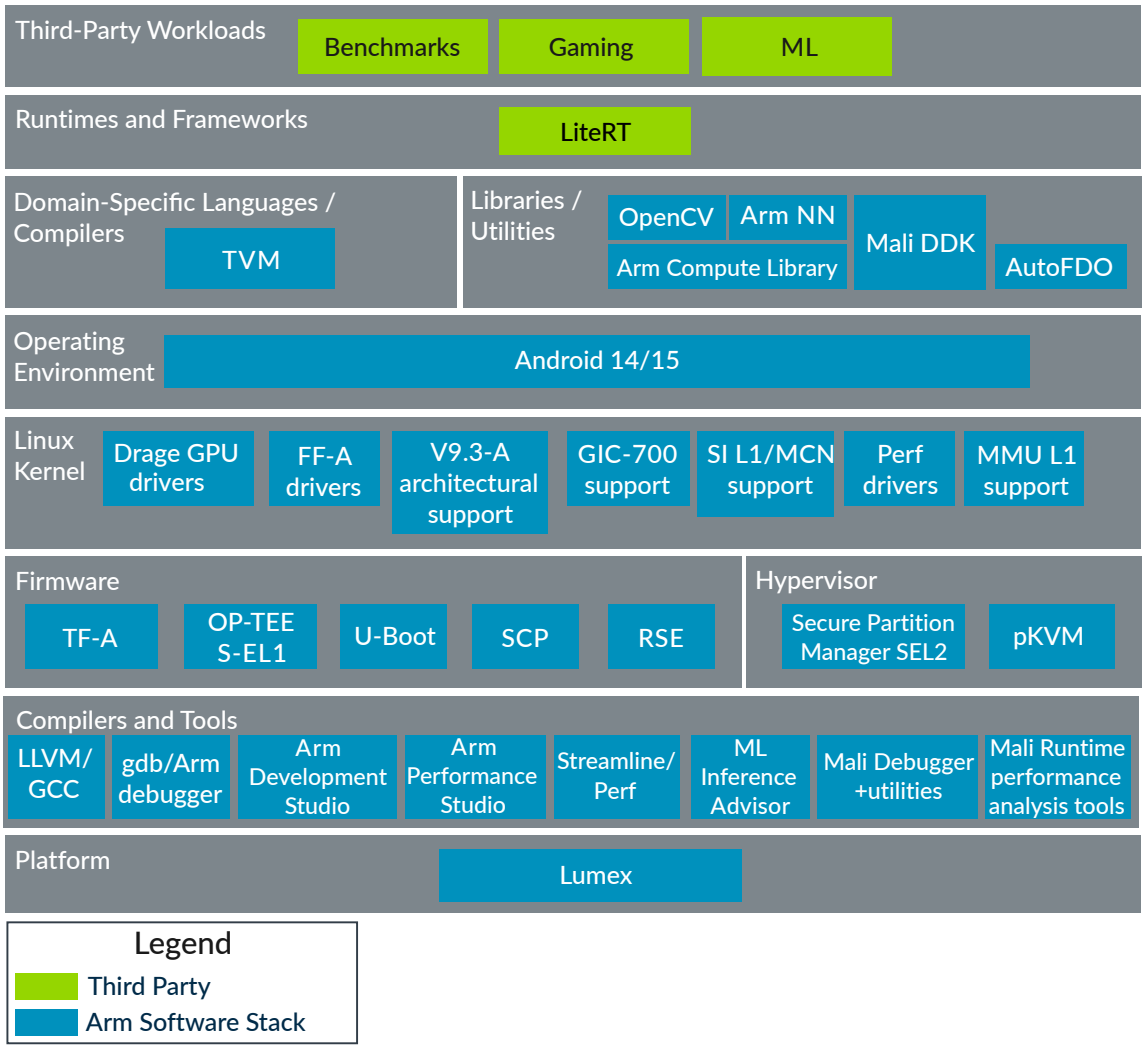
Region	Submodule	Start address (Hex)	End Address (Hex)	Size	Comment
RAM/ROM	dpu_adu	0x00_0000_0000	0x00_000F_FFFF	1 MB	AP Secure Boot RAM
RAM/ROM	dpu_gcu	0x00_0010_0000	0x00_001F_FFFF	1 MB	RESERVED for RAM/ROM
RAM/ROM	dpu_tcu_s1	0x00_0020_0000	0x00_002F_FFFF	1 MB	RESERVED for RAM/ROM
RAM/ROM	dpu_dsi	0x00_0030_0000	0x00_003F_FFFF	1 MB	RESERVED for RAM/ROM
RAM/ROM	RESERVED	0x00_0040_0000	0x00_004F_FFFF	1 MB	RESERVED for RAM/ROM
RAM/ROM	ela	0x00_0050_0000	0x00_0050_FFFF	64 KB	RESERVED for RAM/ROM
RAM/ROM	rsvd	0x00_0051_0000	0x00_007F_FFFF	2.9 MB	RESERVED for RAM/ROM

3. Introduction to the Lumex software stack

The software stack includes open-source code available from the relevant upstream projects, including *System Control Processor* (SCP) firmware, Trusted firmware, Linux kernel, Android, Arm NN, and more.

The following figure shows all the components of the software stack.

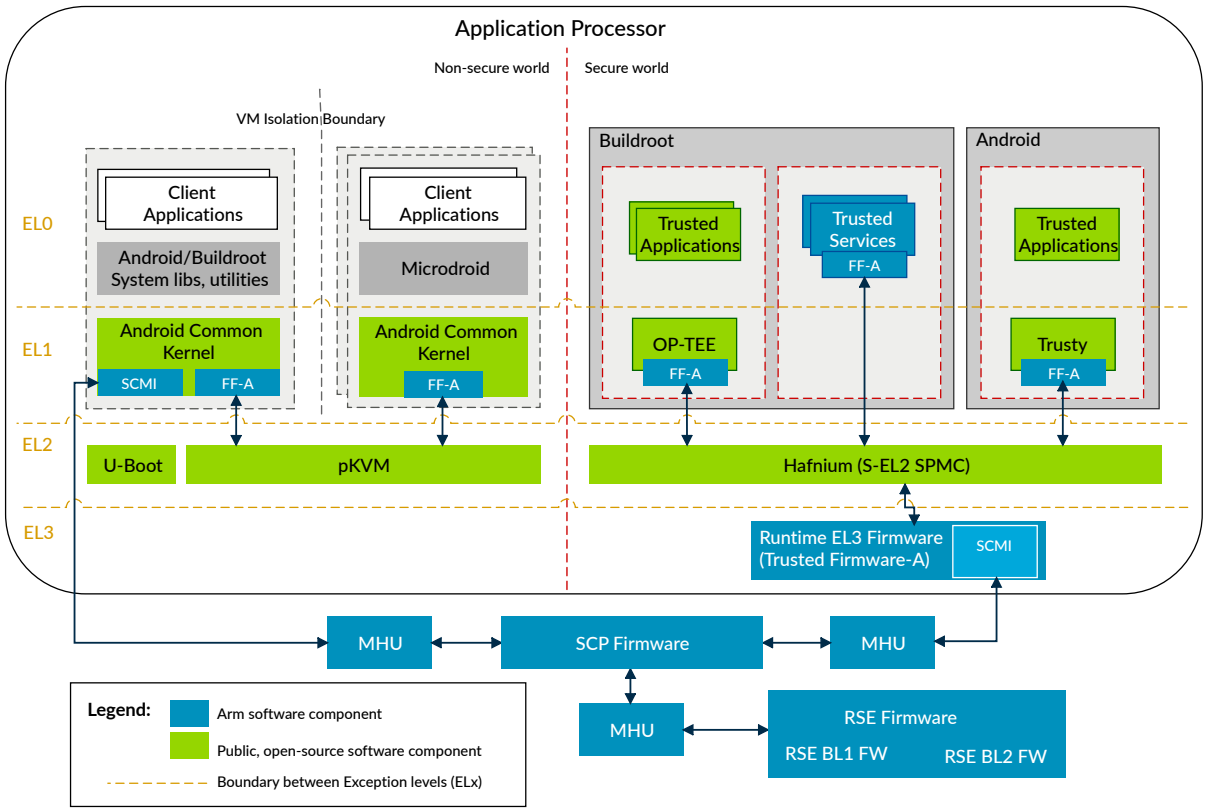
Figure 3-1: Components of Lumex software stack



The software stack is developed and validated against the Lumex FVP and proven on Arm internal hardware emulation environments.

The following figure shows a runtime view of the software stack at the AArch64 Exception levels: EL0, EL1, EL2, and EL3. The higher-numbered Exception levels have a higher level of privilege.

Figure 3-2: Runtime view of Lumex software stack



For more information on the Lumex software stack, instructions on how to set it up, and how to run it on the FVP, see Building the software stack.

3.1 RSE firmware

The Runtime Security Engine (RSE-100) is the Root of Trust for the Lumex platform on FVP.

For detailed information, see the *Trusted Firmware-M documentation*.

Note

In Arm documentation, the RSE is sometimes referred to as the Runtime Security Subsystem (RSS).

The Lumex boot sequence is as follows:

1. RSE Boot Loader stage 1 (BL1) code executes right after a cold reset or powerup.

2. RSE initially boots from immutable code (BL1_1) in its internal ROM, then continues with BL1_2 that is provisioned and hash-locked in RSE One-Time Programable (OTP) memory.
3. The updatable MCUboot BL2 boot stage loads from the flash into RSE SRAM, where it is authenticated.
4. BL2 loads and authenticates the TF-M runtime firmware into RSE SRAM from the host flash. BL2 is also responsible for loading initial boot code into other subsystems within Lumex:
    - SCP firmware
    - AP BL1

3.2 SCP firmware

The System Control Processor (SCP) is responsible for low-level system management. The SCP is a Cortex®-M85 processor with a set of dedicated peripherals and interfaces that you can extend.

SCP firmware supports:

- Power-up sequence and system start-up
- Initial hardware configuration
- Clock management
- Servicing power state requests from the OS Power Management (OSPM) software

SCP firmware performs the following functions:

- Sets up the generic timer, UART console, and clocks
- Powers on the primary AP
- Responds to SCMI messages using MHUs for processor core power control and Dynamic Voltage Frequency Scaling (DVFS)
- Power Domain management
- Clock management

3.3 AP Secure world software

Secure software and firmware runs in the AP Secure world. It consists of AP firmware, Secure Partition Manager, and Secure Partitions (OP-TEE, Trusted Services, and Trusty).

Trusted Firmware for A profile

The AP firmware consists of the code that is required to boot the Lumex platform up to the point where the Rich Operating System (ROS) execution starts. This firmware performs architecture and platform initialization. AP firmware also loads and initializes Secure world images like Secure Partition Manager and Trusted OS.

For detailed information, see the *Trusted Firmware-A documentation*.

Trusted Firmware-A BL1

Boot Loader stage 1 (BL1) performs minimal architectural initialization for example exception vectors, processor core initialization, and platform initialization. It loads the BL2 image and passes control to it.

Trusted Firmware-A BL2

Boot Loader stage 2 (BL2) runs at S-EL1 and performs architectural initialization required for subsequent stages of TF-A and normal world software. It configures the Memory Side Filter (MSF) and isolates memory regions in DRAM for Secure and Non-secure use.

BL2 loads and authenticates the following images in sequence:

1. EL3 Runtime Software (BL31 image)
2. Secure Partition Manager and Secure Partition images (BL32 image), containing Hafnium, OP-TEE, Trusted Services, and Trusty
3. Non-Trusted firmware - U-Boot (BL33 image)

Trusted Firmware-A BL31

BL2 loads and authenticates EL3 Runtime Software (BL31) and BL1 passes EL3 control to BL31. In Lumex, BL31 runs at trusted SRAM.

BL2 provides the following runtime services:

- Power State Coordination Interface (PSCI)
- System Control and Management Interface (SCMI)
- Secure Monitor Framework
- Secure Partition Manager Dispatcher

Secure Partition Manager

Lumex enables the FEAT_SEL2 architectural extension, and it uses Hafnium as the Secure Partition Manager Core (SPMC). The `BL32` build option in TF-A specifies the SPMC image. The SPMC component runs at S-EL2 exception level.

Secure Partitions

Software images isolated using Secure Partition Manager (SPM) are referred to as Secure Partitions. Lumex enables Open Portable Trusted Execution Environment (OP-TEE), Trusted Services, and Trusty as Secure Partitions.

OP-TEE

The Open Portable Trusted Execution Environment (OP-TEE) Trusted OS is virtualized using Hafnium at S-EL2. OP-TEE OS for Lumex is built with FF-A and S-EL2 SPMC support. This enables OP-TEE as a Secure Partition running in an isolated address space managed by Hafnium. The OP-TEE kernel runs at S-EL1 with Trusted applications running at S-EL0.

Trusted Services

Trusted Services like Cryptographic Services and Internal Trusted Storage run as S-EL0 Secure Partitions.

Trusty

Trusty is a Secure OS that provides a Trusted Execution Environment (TEE) for Android. Trusty runs as a Secure Partition image in an isolated address space in S-EL1, virtualized and managed by Hafnium running at S-EL2. The Trusty kernel runs at S-EL1 with Trusted applications running at S-EL0.

3.4 AP Non-secure world software

This section describes the AP Non-secure world software. Non-secure world refers to the part of the system that operates without access to secure resources.

U-Boot

TF-A BL31 passes execution control to the Universal Bootloader (U-Boot) at BL33.

U-Boot in Lumex supports multiple image formats:

- The FitImage format contains the Linux kernel and the Buildroot ramdisk, which are authenticated and loaded in their respective positions in DRAM. Execution then passes to the kernel.
- The Android boot image contains the Linux kernel and the Android ramdisk. If you use Android Verified Boot (AVB), the boot image loads itself into DRAM using virtio on the FVP. The boot image is then authenticated and execution passes to the kernel.

Android Common Kernel

The Android Common Kernel in Lumex contains the subsystem-specific features that demonstrate the capabilities of Lumex.

Apart from default configuration, it enables:

- Arm MHUv3 controller driver
- Arm FF-A driver
- OP-TEE driver with FF-A Transport Support
- Arm FF-A user space interface driver
- Trusty driver with FF-A Transport Support
- Virtualization using protected Kernel-based Virtual Machine (pKVM)

SMMU Virtualization

Arm MMU L1 System Memory Management Unit provides virtual-to-physical address translation for connected devices. This gives the appearance of access to the full memory space with the advantages of memory isolation and protection.

For more information on Arm MMU L1 System Memory Management Unit, see *Arm® MMU L1 System Memory Management Unit Technical Reference Manual*.

Linux has two SMMUv3 drivers:

- `CONFIG_ARM_SMMU_V3` enables the normal kernel driver that executes at `EL1`
- `CONFIG_ARM_SMMU_V3_PKVM` enables a split driver that executes partly at `EL2`, in the pKVM hypervisor

When pKVM is enabled (`kvm-arm.mode=protected`), the pKVM SMMU driver takes precedence over the normal driver, and protects hypervisor and guest VMs from host DMA. Host device drivers still configure DMA using the Linux DMA API. After making sure that a compromised host does not use DMA, the hypervisor installs the requested virtual-to-physical translations into the SMMU stage-2 page tables.

Android support

Lumex supports the Android Open-Source Project (AOSP) that contains the following.

- Android framework
- Native Libraries
- Android Runtime and the Hardware Abstraction Layers (HALs) for the Android OS.

The Lumex device profile defines the required variables for Android, for example partition size and product packages, and has support for the following Android configurations:

Software rendering

This profile has support for Android UI and boots to the Android home screen using SwiftShader. Swiftshader is a processor core base implementation of the Vulkan graphics API by Google.

Hardware rendering

This profile also has support for Android UI and boots to the Android home screen. The Mali™ G1-Ultra core is used for rendering.

Microdroid

Microdroid is a lightweight version of Android that runs in a protected Virtual Machine (pVM) and is managed by Android using ChromeOS Virtual Machine Monitor (crosvm).

To run the included Microdroid demo or an actual Microdroid instance, see Run Microdroid.

Buildroot

Buildroot provides a minimal root file system that boots quickly and is useful for testing the Board Support Package (BSP). The interface is text only with no graphics support.

For information about running Buildroot tests, see Buildroot tests.

LiteRT Machine Learning

The Lumex platform provides a minimal CMake wrapper project for building Lite Runtime (LiteRT) applications for Lumex targets. LiteRT was formerly known as TensorFlow Lite. By default, this project builds the `benchmark_model` application that you can use to profile and validate ML inference flows. However, you can easily adapt the project and build any application exposed by LiteRT.

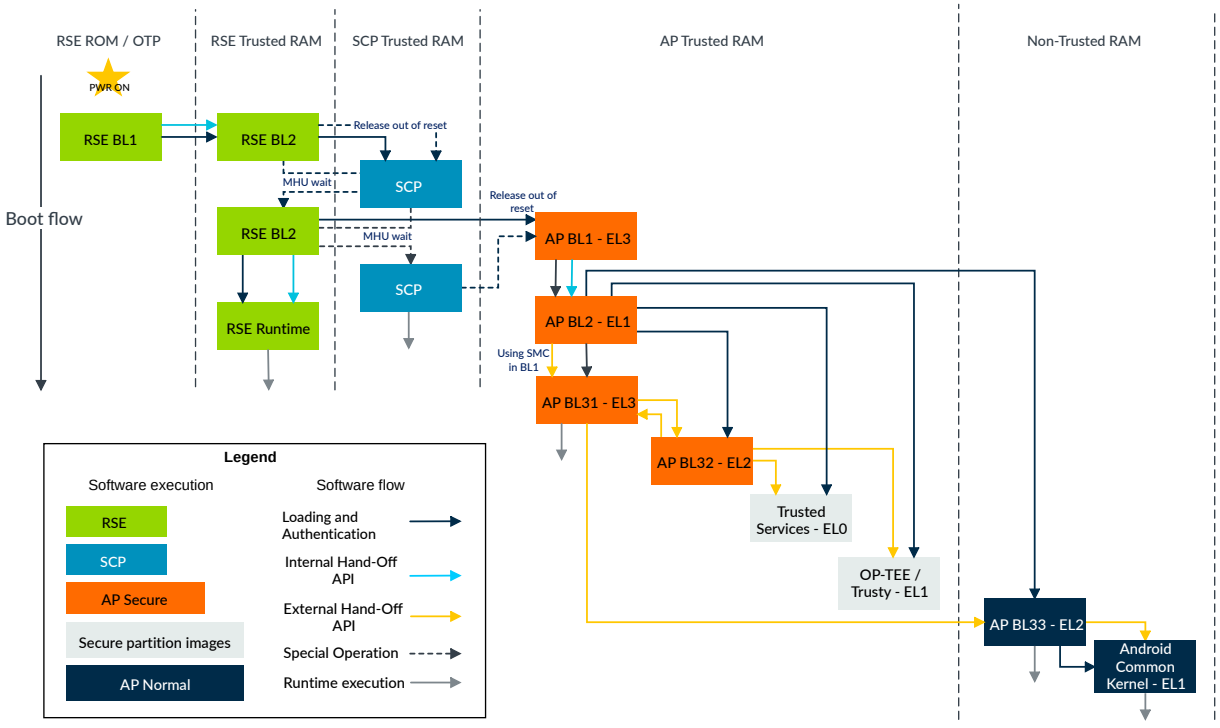
Note

You cannot use LiteRT on the Lumex FVP to generate representative performance numbers. The ML stack integration on the Lumex FVP is for demonstration purposes.

3.5 Boot flow sequence

The following figure illustrates the boot flow sequence in detail.

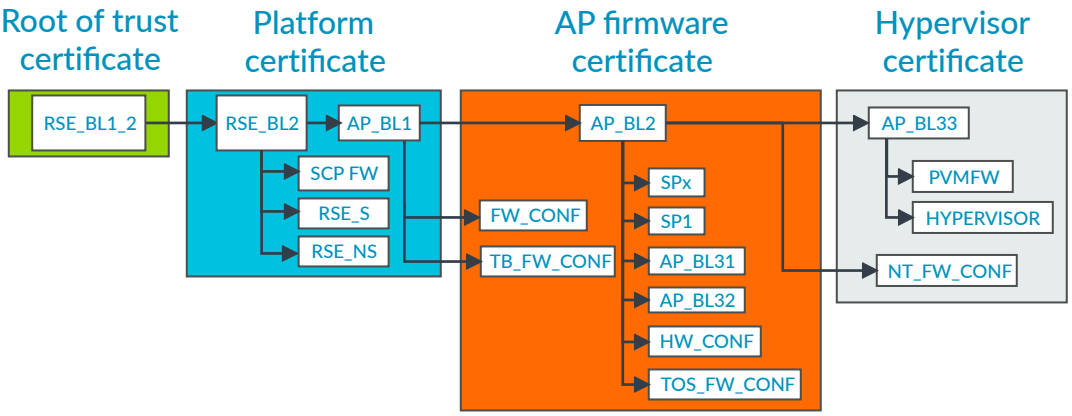
Figure 3-3: Lumex boot flow sequence



The certificates on the Lumex platform FVP are stored in RSE-100. These sample certificates validate the images and you can use them as a basis on which to create your own certificates.

They are loaded in the sequence shown in the following figure:

Figure 3-4: Lumex certificate structure



4. Building the software stack

Before you can use the Lumex software stack, you must set up the compilation environment, download the Lumex software code, and then compile the code.

4.1 Debian OS build options

Currently, the Debian OS based build distribution does not support hardware or software rendering options. Considering this limitation, this build variant should be only used for development or validation work that does not imply pixel rendering.

4.2 Review Android options

For Android distributions, you can choose options for GPU rendering and Android Verified Boot (AVB) according to your requirements.

Android hardware and software rendering options

The Android build distribution supports the following GPU rendering options:

Table 4-1: Android GPU options

TC_GPU value	Description
`swr`	Display with Swiftshader (software rendering)
`hwr-prebuilt`	For Android 15 only, *Mali G1-Ultra* GPU (hardware rendering based on prebuilt binaries)

Android Verified Boot options

You can build the Android images with or without authentication enabled. Authentication uses AVB. To build with AVB, set the AVB variable `export AVB=true` before the build, and use the `-a` option when you run the software on the FVP. The build uses userdebug mode, and the resulting image takes longer to boot because of image verification. This option does not affect the way the system boots, but it adds an optional confidence check on the prerequisite images.

4.3 Set up the build environment

The Lumex software stack uses bash scripts to build a *Board Support Package* (BSP) and a choice of Android, Buildroot, or Debian distributions.

Procedure

1. Ensure that your host computer is running Ubuntu Linux 20.04 or 22.04 (or related releases such as 20.04.n or 22.04.n). For example:

```
$ cat /etc/issue
```

```
Ubuntu 22.04.5 LTS
```

2. Ensure that your host computer meets the requirements for building and running Android, including:
   - At least 400 GB of free disk space to check out and build the code. Increase this amount if you conduct multiple builds.
   - At least 64 GB of available RAM. Lower amounts of RAM can lead to build failures due to lack of memory.
3. Check that bash is the default command shell by entering the command `echo $0` in a terminal window. If the terminal window displays `bash`, skip to the next step, otherwise enter the following command:

```
$ sudo chsh -s /bin/bash <username>
```

4. Install the repo code synchronization tool from Google:

```
mkdir -p ~/bin
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
export PATH=~/bin:$PATH
```

5. Enter the following commands to check that you have git installed with the necessary minimum git config:

```
sudo apt install git
git config --global user.name "<user name>"
git config --global user.email "<email>"
git config --global protocol.version 2
```

6. Install and start Docker:

```
sudo apt install docker.io
# ensure docker service is properly started and running
sudo systemctl restart docker
```

7. Allow a non-root user to manage Docker:

```
sudo usermod -aG docker $USER
newgrp docker
```

Results

Your environment is ready for you to download and build your chosen distribution.

4.4 Download the source code

The commands for downloading the source code differ depending on the distribution.

Before you begin

Prepare the build environment as described in Set up the build environment.

Procedure

1. Define the `REPO_TARGET`, `TC_BRANCH`, and `MANIFEST` environment variables:

```
export TC_BRANCH=refs/tags/Lumex-1
```

- For Buildroot and Debian, enter:

```
export REPO_TARGET=bsp
export MANIFEST=tc4_a15.xml
```

- For Android, enter:

```
export REPO_TARGET=android
```

For Android 14, enter:

```
export MANIFEST=tc4_a14.xml
```

For Android 15, enter:

```
export MANIFEST=tc4_a15.xml
```

2. Create a folder for your workspace (referred to as <TC_WORKSPACE> in these instructions).

```
mkdir <TC_WORKSPACE>
cd <TC_WORKSPACE>
```

3. Sync the source code:

```
repo init -u https://gitlab.arm.com/arm-reference-solutions/arm-reference-
solutions-manifest \
    -m ${MANIFEST} \
    -b ${TC_BRANCH} \
    -g ${REPO_TARGET}
repo sync -j6
```

If running repo synchronization again, you must also rerun the `setup.sh` script because the `repo sync` command can discard patches. See Prepare the compilation environment.

Results

Once the previous process finishes, the <TC_WORKSPACE> directory has the following structure:

build-scripts
: component build scripts

run-scripts
: scripts to run the FVP

src
: each component git repository

tests
: source code for tests

4.5 Prepare the compilation environment

Preparing the compilation environment consists of downloading the cross-compilation toolchain and setting up the Python virtual environment.

The commands in this section are distribution specific and result in:

- Setting up a docker image
- Patching components
- Installing the toolchain and build tools

Before you begin

- Review Android options
- Review Debian options
- Set up the build environment
- Download the source code

Additional Android 14 requirements for hardware rendering

To use Android 14 hardware rendering, you must get additional files from Arm Support. These files contain the minimal DDK support source code, and are available only to licensee partners.

Arm have tested the Lumex software stack using the following instructions and file revisions:

1. Get the following files from Arm Support:
   - `AX504X08X-SW-99005-r48p0-00dev0.tar`
   - `AX504X124-r48p0-00dev0-1-5194ffc9.tar.gz`
   - `ZX002-SW-98000-r48p0-00rel0.tar`
   - `ZX002-SW-98001-r48p0-00rel0.tar`
   - `ZX002-SW-98003-r48p0-00rel0.tar`
   - `ZX002-SW-98004-r48p0-00rel0.tar`
2. Create a folder `hwr` in `<TC_WORKSPACE>`. Make sure you run following command from the `<TC_WORKSPACE>` directory:

   ```
   mkdir -p hwr
   ```

3. Move the downloaded archive files for hardware rendering support into the newly created `hwr` folder. You do not need to extract them.

Android considerations

Fetching and building Android code can take in excess of two hours, depending on your host system hardware specification and network.

Note

The Android *Software Development Kit* (SDK), which is required to build the `benchmark_model` application for Android, has standalone terms and conditions. These terms and conditions are automatically accepted during Android SDK installation process and can be found in *Android Studio Terms and conditions*.

Debian considerations

Currently, the Debian build does not support software or hardware rendering. As such, do not define the value of the `TC_GPU` variable. The Debian build can still be a valuable resource when just considering other types of development or validation work, which do not involve pixel rendering.

Procedure

1. Define the platform:

```
export PLATFORM=tc4
```

2. Define the build type as `buildroot`, `debian`, or `android`. For example:

```
export FILESYSTEM=android
```

3. Define `fvp` as the target. For example:

```
export TC_TARGET_FLAVOR=fvp
```

4. Most builds run in parallel using all the available cores. To change this number, use the following command:

```
export PARALLELISM=<number of cores>
```

5. For Android builds, set the Android version to either `android14` or `android15`. For example:

```
export TC_ANDROID_VERSION=android15
```

6. For Android, set the type of GPU rendering to either `hwr` or `swr`. See Android hardware and software rendering options. For example:

```
export TC_GPU=swr
```

Note

If you are using hardware rendering on Android 14, you must have downloaded the required files described in Additional Android 14 requirements for hardware rendering

7. For hardware rendering on Android, set the following variables:

```
# following variable assumes hardware rendering support files are located in
 "<TC_WORKSPACE>/hwr"
# if this is not the case amend it accordingly
```

```
export TC_ANDROID_DDK_PDH_FILES="$(pwd)/hwr"
export LM_LICENSE_FILE=<LICENSE FILE>
export ARMLMD_LICENSE_FILE=<LICENSE FILE>
```

8. By default, Android builds without AVB. To perform the build with AVB enabled, enter the following command:

```
export AVB=true
```

9. After setting the variables, run the build script:

```
cd build-scripts
./setup.sh
```

Examples

Buildroot build.

```
export PLATFORM=tc4
export FILESYSTEM=buildroot
export TC_TARGET_FLAVOR=fvp
cd build-scripts
./setup.sh
```

Debian build without software or GPU hardware rendering support

```
export PLATFORM=tc4
export FILESYSTEM=debian
export TC_TARGET_FLAVOR=fvp
cd build-scripts
./setup.sh
```

Android build with hardware rendering support based on prebuilt binaries.

```
export PLATFORM=tc4
export FILESYSTEM=android
export TC_ANDROID_VERSION=android15
export TC_GPU=hwr-prebuilt
export TC_TARGET_FLAVOR=fvp
cd build-scripts
./setup.sh
```

Android with software rendering support.

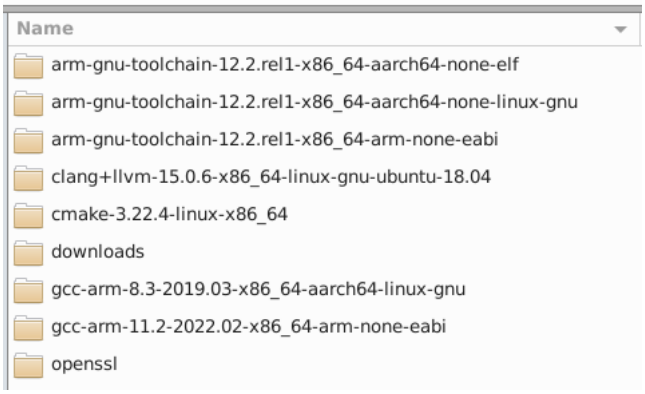
```
export PLATFORM=tc4
export TC_GPU=swr
export TC_TARGET_FLAVOR=fvp
export FILESYSTEM=android
export TC_ANDROID_VERSION=android15
cd build-scripts
./setup.sh
```

If building the Lumex software stack for more than one target, please ensure you run a clean build between each different build to avoid setup and building errors. See Build the software stack images for more information about running clean builds.

Results

The cross-compilation toolchain is present in the `<TC_WORKSPACE>/tools/` directory, and the Python virtual environment is present in the `<TC_WORKSPACE>/tc-venv/` directory. For example:

Figure 4-1: Tools directory



4.6 Build the software stack images

The `build-all.sh` script builds all the components. Each component also has its own script, allowing it to be built, cleaned, and deployed separately. All scripts support the `build`, `clean`, `deploy`, and `patch` commands. The `build-all.sh` script also supports the `all` command to clean and rebuild the entire stack. When building the software stack for more than one target, run a clean build between each build to avoid errors.

Before you begin

- Set up the build environment
- Download the source code
- Prepare the compilation environment

Procedure

1. To build the entire software stack for any of the supported distributions enter the following command:

   ```
   ./run_docker.sh ./build-all.sh build
   ```

   The output directory for the build, referred to in this guide as `<TC_OUTPUT>`, is located in this directory:

   ```
   <TC_WORKSPACE>/output/<$PLATFORM>/<$FILESYSTEM>/<$TC_TARGET_FLAVOR>/<$TC_GPU>
   ```

For example,

```
<TC_WORKSPACE>/output/tc4/buildroot/fvp/swr/
```

For Buildroot and Debian distributions, the <$TC_GPU> option defaults to swr if not defined.

When the build finishes, <TC_OUTPUT> contains two directories:

- tmp_build/ contains the individual component build files
- deploy/ contains the final images

2. To build a specific component of the software stack use the component-specific script. For example, to clean, build, and deploy SCP, enter the following commands:

```
./run_docker.sh ./build-scp.sh clean
./run_docker.sh ./build-scp.sh build
./run_docker.sh ./build-scp.sh deploy
```

Defining the platform and filesystem is described in Prepare the Compilation Environment, but you can also specify this information as part of the command:

```
./run_docker.sh ./build-all.sh \
    -p $PLATFORM \
    -f $FILESYSTEM \
    -a $AVB \
    -t $TC_TARGET_FLAVOR \
    -g $TC_GPU build
```

3. To modify the dependencies of a specific component and rebuild it, edit the build_requirements.txt file in the build-scripts directory. The dependencies have the format $component=$dependency. Perform the component build using the with_reqs option. This sequentially rebuilds the specified component and its dependencies, using the current environment variables:

```
./run_docker.sh ./build-scp.sh clean build with_reqs
```

Other aspects of the build-*.sh script function as described previously.

Results

<TC_OUTPUT>/deploy/ contains the final images, and <TC_OUTPUT>/tmp_build/ contains the component build files.

4.7 Scripts and file locations

This section lists the software components, their corresponding build script, and the location of the installed files.

The following tables give the script and file locations relative to these directories:

- Scripts: <TC_WORKSPACE>/build-scripts

- Files: `<TC_OUTPUT>`

Firmware components

The following firmware components are provided.

Table 4-2: Provided firmware components, scripts, and files

Component	Script	Files	Description

Trusted Firmware-A	`build-tfa.sh`	`/deploy/bl1-tc.bin`
		
		`/deploy/fip-tc.bin`
		
		`/deploy/fip_gpt-tc.bin`
			Trusted Firmware-A documentation

Runtime Security Engine (RSE)	`build-rse.sh`	`/deploy/rss_rom.bin`
		
		`/deploy/rse_encrypted_cm_provisioning_bundle_0.bin`
		
		`/deploy/rse_encrypted_dm_provisioning_bundle_0.bin`
			RSE Firmware

System Control Processor (SCP)	`build-scp.sh`	`/deploy/scp-css.bin`	SCP Firmware
U-Boot	`build-u-boot.sh`	`/deploy/u-boot.bin`	*U-boot*
Hafnium	`build-hafnium.sh`	`/deploy/hafnium.bin`	*Hafnium*
OP-TEE	`build-optee-os.sh`	`/tmp_build/tfa_sp/tee-pager_v2.bin`	*OP-TEE*

S-EL0 trusted-services	`build-trusted-services.sh`	`/tmp_build/tfa_sp/crypto.bin`
		
		`/tmp_build/tfa_sp/internal-trusted-storage.bin`
		
		`/tmp_build/tfa_sp/fwu.bin`
			Trusted Services

Linux (for building the ACK)	`build-linux.sh`	`/deploy/Image`	*Android Common Kernel (ACK)*
Trusty	`build-trusty.sh`	`/tmp_build/tfa_sp/lk.bin`	*Trusty*
TensorFlow	`build-ml-app.sh`	`/deploy/benchmark_model`	*TensorFlow*

Distributions

The following distributions are provided. The Buildroot Linux distribution is based on BusyBox and built using `glibc`.

Table 4-3: Provided distributions, scripts, and files

Component	Script	Files	Description
Buildroot Linux	`build-buildroot.sh`	`/deploy/tc-fitImage.bin`	*Buildroot GitHub*

Debian Linux	`build-debian.sh`	`/deploy/debian.img`
		
		`/deploy/debian_fs.bin`
			Debian GitHub

Component	Script	Files	Description

Android	`build-android.sh`	`/deploy/android.img`
		
		`/deploy/ramdisk_uboot.img`
		
		`/deploy/system.img`
		
		`/deploy/userdata.img`
			Android documentation

		`/deploy/boot.img`	AVB only
		`/deploy/vbmeta.img`	AVB only

Run scripts

You can run the software using scripts in the `<TC_WORKSPACE>/run-scripts/` directory.

For more information, see Running the software on the FVP.

5. Running the software on the FVP

You can run the Lumex software using either Android, Buildroot, or Debian on the Lumex FVP.

5.1 Launch the FVP

Launch the FVP as described in this section using the provided script.

Procedure

1. Download the Lumex FVP version 11.29.51 from Arm Developer. For more information about the FVP platform, see the `release_note.txt` file in the FVP bundle.
2. Run the FVP to ensure that all dependencies are met:

   ```
   ./path/to/FVP_RD_Lumex
   ```

   The FVP launches, displaying a graphical interface showing information about the current state of the FVP and four terminal windows. Press Ctrl+C to close the FVP and continue with the next step.
3. Launch the FVP using the `run_model.sh` script in `<TC_WORKSPACE>/run-scripts/tc4`, providing the previously built images as arguments. The script has the following options:

Table 5-1: Script options

Option	Description	Required or Optional	Default
`-m, --model MODEL`	Path to model	Required	-
`-d, --distro {buildroot | android | debian}`	Distribution type	Required	-
`-a, --avb {true | false}`	Android Verified Boot (AVB)	Optional	`false`
`-t, --tap-interface`	TAP interface	Optional	-
`-n, --networking {user | tap | none}`	Networking	Optional	`tap` if TAP interface provided, otherwise `user`
`--debug {iris | cadi | none}`	Start a debug server, print the listening port, and wait for the debugger	Optional	`none`
`-v, --no-visualisation`	Do not spawn a model visualization window	Optional	-
`--telnet`	Do not spawn console windows, only listen on telnet	Optional	-
`-- MODEL_ARGS`	Pass all further options directly to the model	Optional	-

- To run Buildroot, enter the following command:

  ```
  ./run-scripts/tc4/run_model.sh -m <model binary path> -d buildroot
  ```

- To run Debian, enter the following command:

```
./run-scripts/tc4/run_model.sh -m <model binary path> -d debian
```

- To run Android without AVB, enter the following command:

```
./run-scripts/tc4/run_model.sh -m <model binary path> -d android
```

- To run Android with AVB enabled, enter the following command:

```
./run-scripts/tc4/run_model.sh -m <model binary path> -d android -a true
```

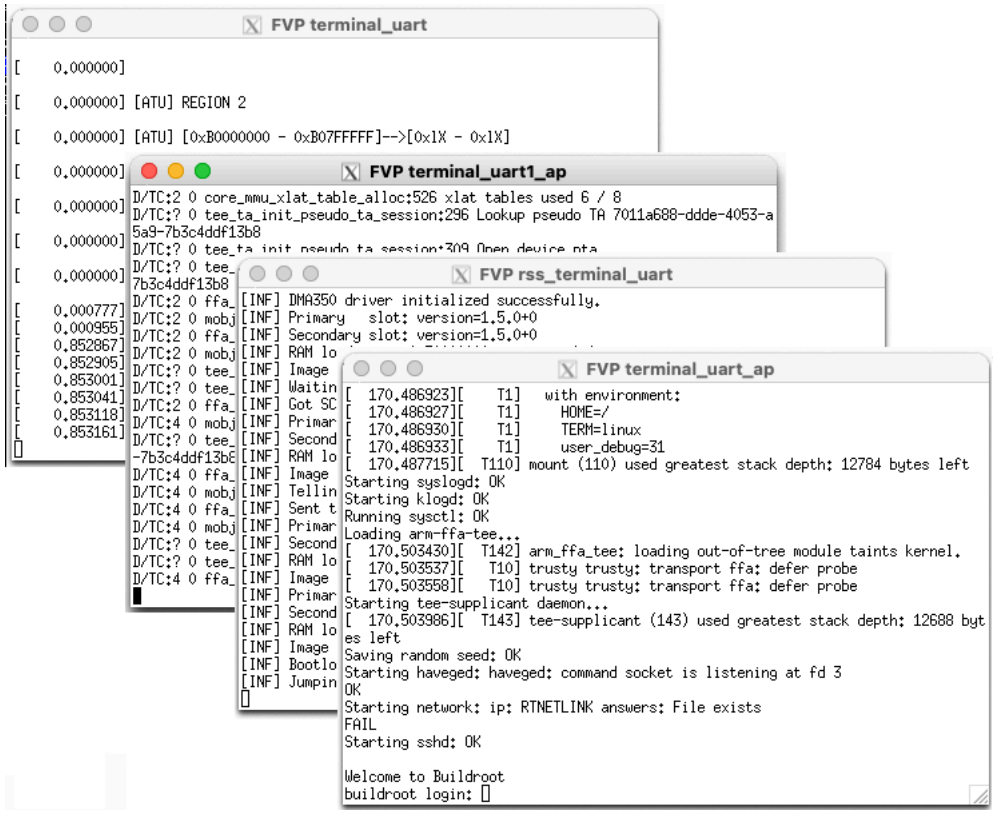
Results

The script runs and displays four terminal windows stacked on top of each other.

Note

- After Buildroot or Debian boots, press Enter to display the Linux login prompt on the **terminal_uart_ap** window. Log in with the username `root`. For Debian you must also provide the password `root`.
- After Android boots, press Enter to display the Android console on the **terminal_uart_ap** window.
- When booting Android, the GUI window **RD Lumex DP0** displays the Android logo while booting and when it finishes, the Android home screen.
- Booting Android can take in excess of an hour, depending on your host system hardware specification and network.

Figure 5-1: Software stack terminal windows



- **terminal_uart_ap** used by the Non-secure world components U-Boot, Android Common Kernel, and filesystem (Android, Buildroot, and Debian)
- **terminal_uart1_ap** used by the Secure world components TF-A, Hafnium, Trusty, and OP-TEE
- **terminal_uart** used for the SCP logs
- **rss_terminal_uart** used by RSE logs

Once the FVP is running, the hardware Root of Trust does the following:

- Verifies the AP and SCP images
- Initializes various crypto services
- Hands over execution to the SCP

The SCP brings the AP out of reset. The AP starts booting from its ROM and boots the following:

- Trusted Firmware-A
- Hafnium
- Secure Partitions (OP-TEE, Trusted Services in Buildroot, and Trusty in Android)
- U-Boot
- The root filesystem of the corresponding Android Common Kernel distribution.

6. Setting up debugging

Describes how to set up debugging, collect trace information, and enable various logs.

6.1 Debug the software stack with Arm Development Studio

The steps in this section require Arm™ Development Studio Platinum.

Arm Development Studio Platinum is only available to licensee partners. For access contact Arm.

6.1.1 Import the FVP

Before you can begin debugging, you must import the Lumex FVP into Arm Development Studio.

Before you begin

Make sure that you have built the target distribution with debug enabled. Debug is enabled by default, but you can configure the file `<TC_WORKSPACE>/build-scripts/config/common.config` to enable debug by setting `TFA_DEBUG=1`.

Procedure

1. Run the FVP as described in Running the software on the FVP with the extra parameters `-- -I` to attach to the debugger. For example:

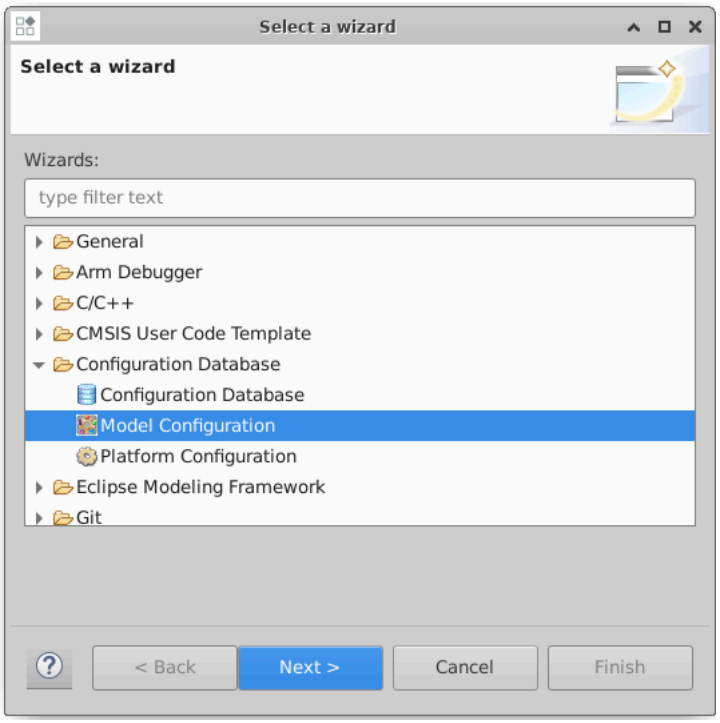
   ```
   ./run-scripts/tc4/run_model.sh -m <model binary path> -d <distro> --debug {iris |
    cadi} -- -I
   ```

   The FVP startup messages display the port number of the Iris server. For example:

   ```
   Iris server started listening to port 7100
   ```

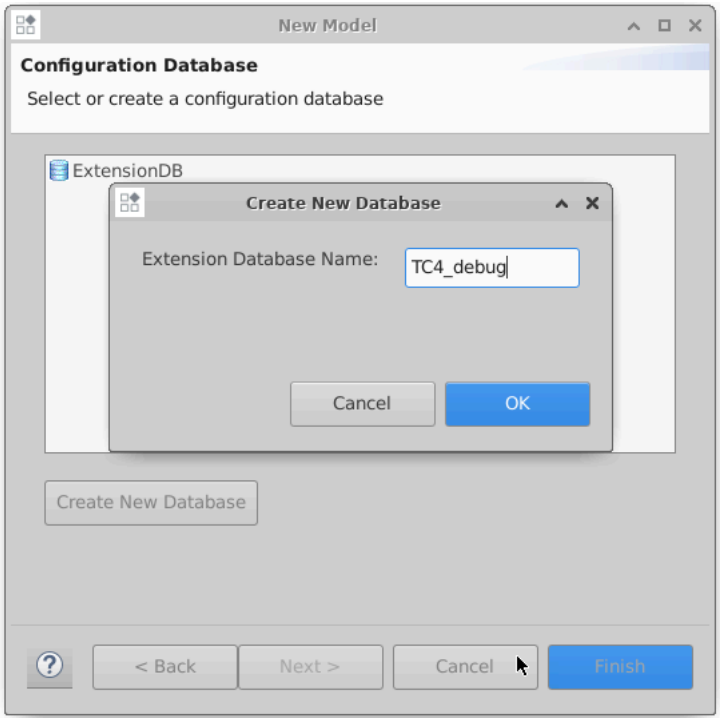
   Make note of the port number, you need it to create debug configurations.
2. Launch Arm Development Studio.
3. In the main menu, select **File** > **New**.
4. Expand the **Configuration Database** group, select **Model Configuration**, and click **Next**.

Figure 6-1: Selecting Model Configuration in Arm Development Studio



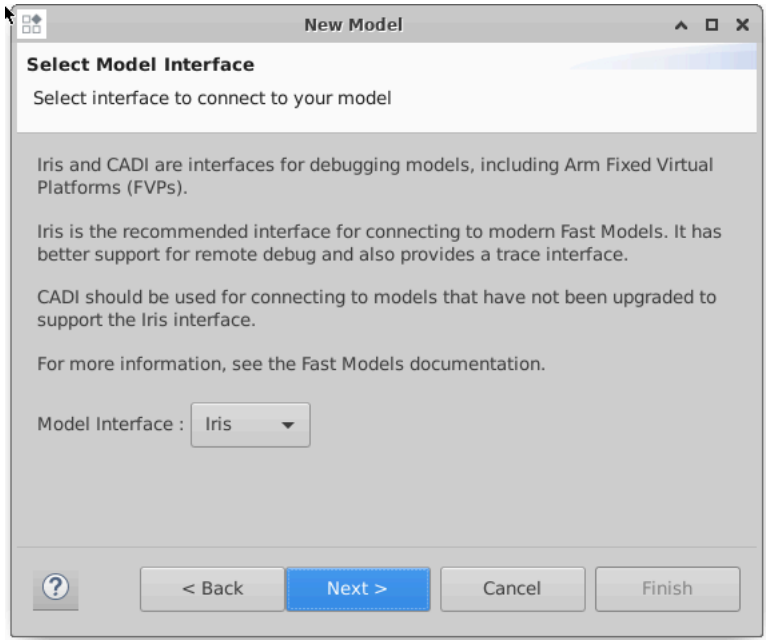
5. Click **Create New Database**, provide a name, and click **OK** to save it.

Figure 6-2: Clicking Create Database in Arm Development Studio



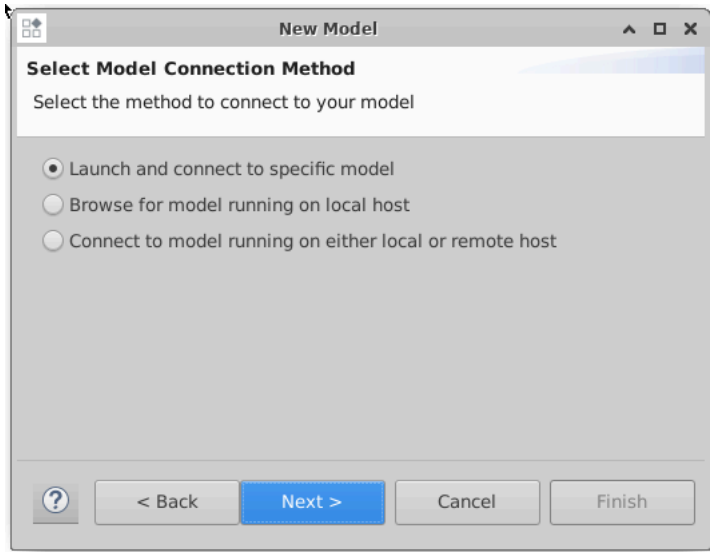
6. Click **Next**. This displays the **Select Model Interface** dialog.

Figure 6-3: Selecting the Model Interface in Arm Development Studio



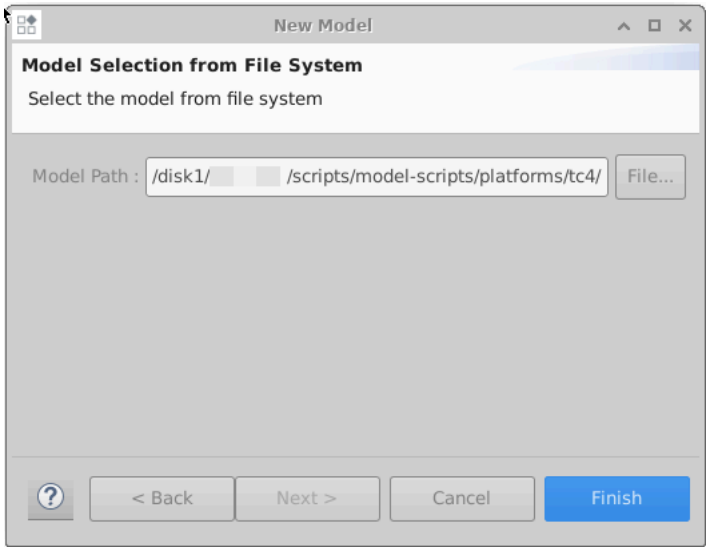
7. Select **Iris** and click **Next**.
8. Select **Launch and connect to specific model** and click **Next**.

Figure 6-4: Selecting the Connection method in Arm Development Studio



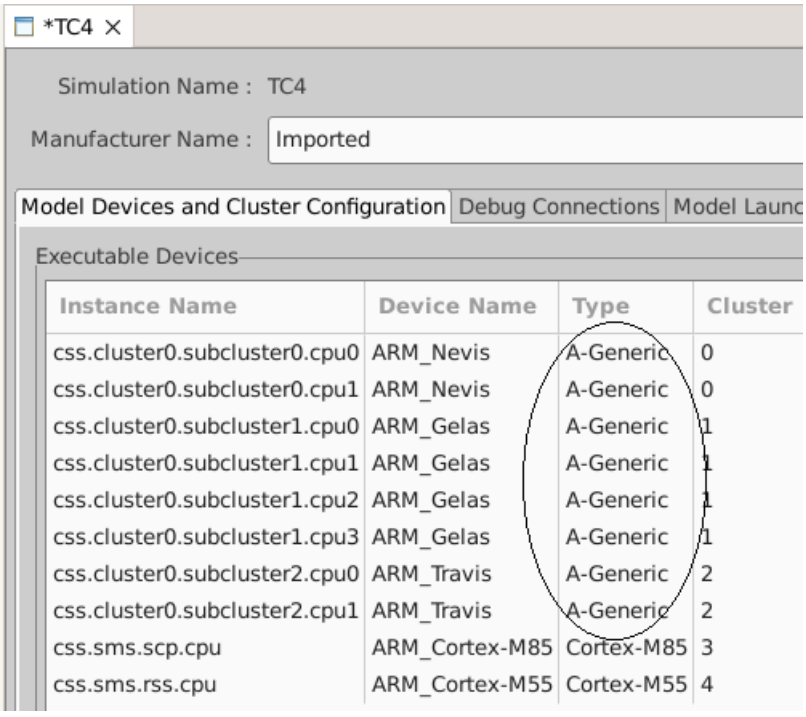
9. Select the **Model Path** and click **Finish**.

Figure 6-5: Selecting the Model Path in Arm Development Studio



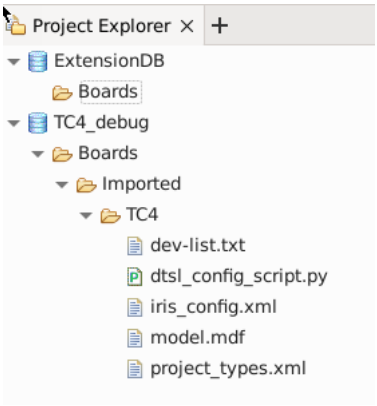
10. The **Model Configuration Editor** loads. If the AP processor cores are not recognized, do the following for each unrecognized processor core:
    a. Select and edit the row.
    b. Select **A-generic** from the list and click **OK**.

Figure 6-6: Changing the processor core type in Arm Development Studio



11. Click **Import**. You can see the imported FVP in **Project Explorer**.

Figure 6-7: Imported FVP in Arm Development Studio



6.1.2 Create debug connections

After importing the Lumex FVP, create debug connections for the AP, RSE, and SCP.

Before you begin

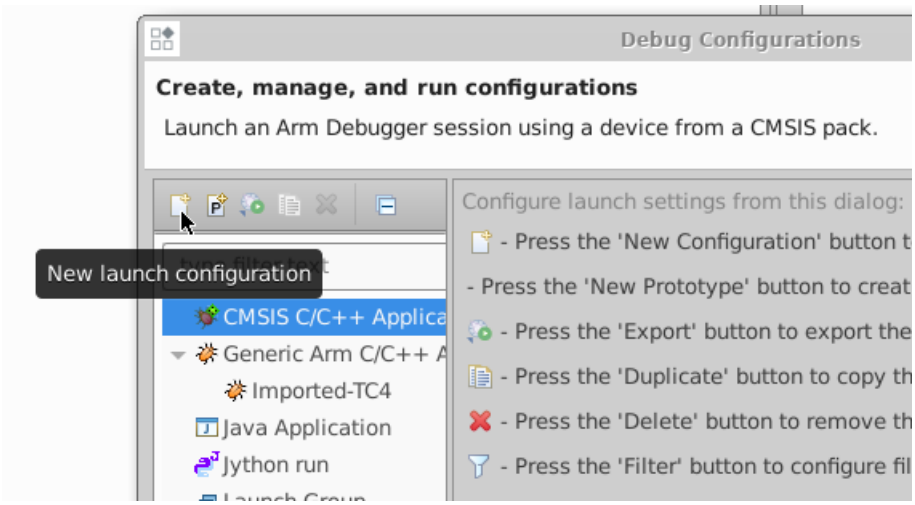
Before you can create a debug configuration, import the FVP into Arm Development Studio.

Import the FVP

Procedure

1. In the main menu, select **Run** > **Debug Configurations**.
2. In the **Create, manage, and run configurations** dialog, select **Generic Arm C/C++ Application**, and click **New launch configuration**.

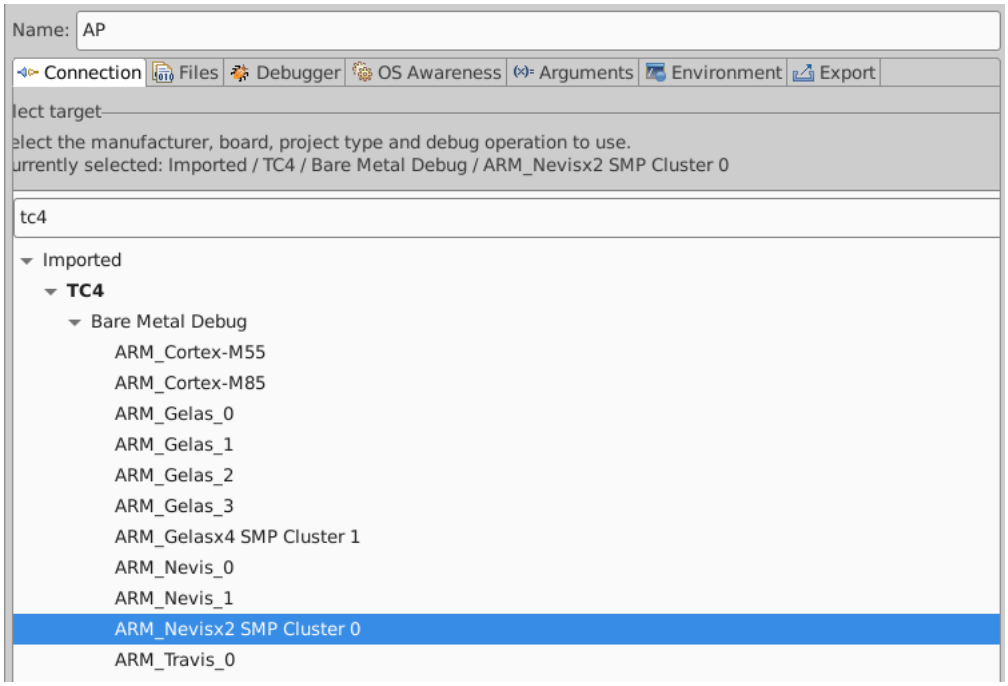
Figure 6-8: Creating launch configuration in Arm Development Studio



3. Select the FVP that you imported in the previous section.

4. Create a debug configuration for the AP with the following properties:

Figure 6-9: AP debug configuration



   a. Ensure that you have selected the desired processor core.
   b. Select **Connect to an already running model**
   c. Specify the **Connection address** as `127.0.0.1:<port-number>`. The `port-number` must match the one for the Iris server displayed on FVP startup. For example, `127.0.0.1:7100`.
   d. Click the **Debug** tab to add debug symbols to the **Execute debugger commands** section of the dialog, and click **Apply**. Alternatively, you can use the **Files** tab to add the path to the ELF files.
   e. Click **Apply** then **Close** to save the debug configuration.
5. Repeat the previous step to create debug configurations for the RSE and SCP. Select `ARM_Cortex-M55` for the RSE and `ARM_Cortex-M85` for the SCP.
6. In the **Debug control** console, select the desired debug target, right-click and select **Connect to Target**.

Results

You can now use the debugging capabilities of Arm Development Studio:

- Use options in the **Debug control** console or keyboard shortcuts to step, run, or halt.
- Use options in the **Command** window for example breakpoints, variable watch, and memory view.

For more information, see the *Arm® Development Studio User Guide*.

6.1.3 Collect FVP trace information

You can run the FVP with trace information enabled for debug and troubleshooting purposes.

About this task

To show the trace output for different stages, the following command examples use the Arm MMU L1 System Memory Management Unit block component. You can adapt the commands for other components as needed.


Note

To collect FVP trace information, you must execute the FVP with the `GenericTrace.so` or `ListTraceSources.so` plugins that are provided and part of your FVP bundle. For more information, see ListTraceSources and GenericTrace in the *Fast Models Reference Guide*.

Procedure

1. List the trace sources:

```
<fvp-model binary path>/FVP_RD_Lumex \
    --plugin <fvp-model plugin path/ListTraceSources.so> \
    >& /tmp/trace-sources-fvp-tc4.txt
```

   The model starts and uses the `ListTraceSources.so` plugin to write the list to a file. The file size can extend to tens of megabytes because the list is extensive.

   The following excerpt shows example output for the component Arm MMU L1 System Memory Management Unit:

```
Component (1556) providing trace: TC4.css.smmu (MMU_1, 11.29.51)
================================================================================
Component is of type "MMU_1"
Version is "11.29.51"
#Sources: 294

Source ArchMsg.Error.error (These messages are about activity occurring on the
 SMMU that is considered an error.
Messages will only come out here if parameter all_error_messages_through_trace is
 true.

DISPLAY %{output})
   Field output type:MTI_STRING size:0 max_size:120 (The stream output)

Source ArchMsg.Error.fetch_from_memory_type_not_supporting_httu (A descriptor
 fetch from an HTTU-enabled translation regime to an unsupported
memory type was made.  Whilst the fetch itself may succeed, if an update to
the descriptor was attempted then it would fail.)
```

2. Run the FVP with trace information enabled:

```
./run-scripts/tc4/run_model.sh -m <model binary path> -d <distro> \
    -- \
    --plugin <fvp-model plugin path/GenericTrace.so> \
    -C 'TRACE.GenericTrace.trace-
sources="TC4.cpnss.smmu_rp0_tcu.*,TC4.css.smmu.*"' \
    -C TRACE.GenericTrace.flush=true
```

To request multiple trace sources, separate the trace-sources with commas as shown in the previous example.

3. By default, the trace information displays on the standard output, for example the screen display. To avoid information scrolling off the screen, redirect the trace information to a file using the following command:

```
./run-scripts/tc4/run_model.sh -m <model binary path> -d <distro> \
    -- \
    --plugin <fvp-model plugin path/GenericTrace.so> \
    -C 'TRACE.GenericTrace.trace-
sources="TC4.cpnss.smmu_rp0_tcu.*,TC4.css.smmu.*"' \
    -C TRACE.GenericTrace.flush=true \
    >& /tmp/trace-fvp-tc4.txt
```

Trace information output can generate large amounts of information depending on the component and filtering options. If redirected to a file, this output can in a short time, result in files that are many gigabytes or even terabytes in size.

The following example shows the trace information captured for the DPU (`streamid=0x00000000`) and GPU (`streamid=0x00000200`):

```
.
.
.
cpnss.smmu_rp0_tcu.start_ptw_read: trans_id=0x0000000000000079
 streamid=0x00000000 substreamid=0xffffffff ttb_grain_stage_and_level=0x00000202
 pa_address=0x000000088ea5bfe0 input_address=0x00000000ff800000 ssd_ns=ssd_ns
 ns=bus-ns desckind=el2_or_st2_aarch64 inner_cache=rawaWB outer_cache=rawaWB
 aprot=DNP adomain=ish mpam_pmg_and_partid=0x00000000 ssd=ns pas=ns
 mecid=0xffffffff
cpnss.smmu_rp0_tcu.verbose_commentary: output="Performing a Table Walk read as:-"
cpnss.smmu_rp0_tcu.verbose_commentary: output="    trans_id:121-st2-final-l2-
aa64-ttb0-vmid:0-ns-sid:0"
cpnss.smmu_rp0_tcu.verbose_commentary: output="to ns-0x000000088ea5bfe0-PND-
u0x5300000a-m0xffffffff-ish-osh-rawaC-rawaC of size 8B"
cpnss.smmu_rp0_tcu.verbose_commentary: output="Table Walk finished:-"
cpnss.smmu_rp0_tcu.verbose_commentary: output="    trans_id:121-st2-final-l2-
aa64-ttb0-vmid:0-ns-sid:0"
cpnss.smmu_rp0_tcu.verbose_commentary: output="got:-"
cpnss.smmu_rp0_tcu.verbose_commentary: output="    0x000000088ea5bfe0:
 0x000000088f2006d5"
cpnss.smmu_rp0_tcu.ptw_read: trans_id=0x0000000000000079 streamid=0x00000000
 substreamid=0xffffffff ttb_grain_stage_and_level=0x00000202
 pa_address=0x000000088ea5bfe0 input_address=0x00000000ff800000 ssd_ns=ssd_ns
 ns=bus-ns desckind=el2_or_st2_aarch64 inner_cache=rawaWB outer_cache=rawaWB
 aprot=DNP adomain=ish abort=ok data=0x000000088f2006d5 ssd=ns pas=ns
 mecid=0xffffffff
cpnss.smmu_rp0_tcu.ptw_read_st2_leaf_descriptor: trans_id=0x0000000000000079
 streamid=0x00000000 substreamid=0xffffffff ttb_grain_stage_and_level=0x00000202
 pa_address=0x000000088ea5bfe0 input_address=0x00000000ff800000
 ssd_ns=ssd_ns ns=bus-ns desckind=el2_or_st2_aarch64 XN=N contiguous=N
 AF=Y SH10=sh10_osh DBM=N HAP21=hap21_read_write MemAttr3_0=memattr_oNC_iNC
 output_address=0x000000088f200000 nT=N s2hwu_pbha=0x00 NS=n/a AMEC=MEC not
 supported. ssd=ns pas=ns mecid=0xffffffff PIE_PIIndex=0xffff PIE_Dirty=n/a
 POE_POIndex=0xffff AssuredOnly=n/a
.
.
.
css.smmu.start_ptw_read: trans_id=0x0000000000000040 streamid=0x00000200
 substreamid=0xffffffff ttb_grain_stage_and_level=0x00000201
```

```
 pa_address=0x0000000883794110 input_address=0x00000008899ad000 ssd_ns=ssd_ns
 ns=bus-ns desckind=el2_or_st2_aarch64 inner_cache=rawaWB outer_cache=rawaWB
 aprot=DNP adomain=ish mpam_pmg_and_partid=0x00000000 ssd=ns pas=ns
 mecid=0xffffffff
css.smmu.verbose_commentary: output="Performing a Table Walk read as:-"
css.smmu.verbose_commentary: output="    trans_id:64-st2-final-l1-aa64-ttb0-
vmid:1-ns-sid:512"
css.smmu.verbose_commentary: output="to ns-0x0000000883794110-PND-u0x53000109-
m0xffffffff-ish-osh-rawaC-rawaC of size 8B"
css.smmu.verbose_commentary: output="Table Walk finished:-"
css.smmu.verbose_commentary: output="    trans_id:64-st2-final-l1-aa64-ttb0-
vmid:1-ns-sid:512"
css.smmu.verbose_commentary: output="got:-"
css.smmu.verbose_commentary: output="    0x0000000883794110: 0x00000008899aa003"
css.smmu.ptw_read: trans_id=0x0000000000000040 streamid=0x00000200
 substreamid=0xffffffff ttb_grain_stage_and_level=0x00000201
 pa_address=0x0000000883794110 input_address=0x00000008899ad000 ssd_ns=ssd_ns
 ns=bus-ns desckind=el2_or_st2_aarch64 inner_cache=rawaWB outer_cache=rawaWB
 aprot=DNP adomain=ish abort=ok data=0x00000008899aa003 ssd=ns pas=ns
 mecid=0xffffffff
css.smmu.ptw_read_st2_table_descriptor: trans_id=0x0000000000000040
 streamid=0x00000200 substreamid=0xffffffff ttb_grain_stage_and_level=0x00000201
 pa_address=0x0000000883794110 input_address=0x00000008899ad000 ssd_ns=ssd_ns
 ns=bus-ns desckind=el2_or_st2_aarch64 APTable=aptable_no_effect XNTable=N
 PXNTable=N TableAddress=0x00000008899aa000 ssd=ns pas=ns mecid=0xffffffff AF=N/A
.
.
.
```

6.1.4 Enabling and capturing Tarmac trace logs

Arm Tarmac Trace Utilities allow you to analyze and browse Tarmac format log files. Tarmac is a human-readable log file format that records timestamped activity in the processor.

Obtain the Arm Tarmac Trace Utilities from the *Tarmac Trace GitHub Repository*.

The documentation in the repository explains how to use the Arm Tarmac Trace Utilities as well as the *Tarmac Trace Community Blog*.

7. Using the Lumex software

Describes the features of Lumex, which vary depending on the distribution.

These features include:

- Optimizing compilation with AutoFDO
- Using ADB connections to the FVP
- Configuring the rotational scheduler
- Setting up a TAP interface
- Updating the firmware

7.1 Optimizing compilation with AutoFDO

Feedback Directed Optimization (FDO), also known as *Profile Guided Optimization* (PGO), uses the program execution profile to guide compiler optimization.

Before you begin

- Compile the application with sufficient debug information to map instructions back to source lines. To do this for Clang and LLVM, add the `-fdebug-info-for-profiling` and `-gline-tables-only` compiler options.
- Use the compiler flag `-Wl,--build-id=sha1` during linking. This flag causes Simpleperf tool to identify the active program or library using the build identifier stored in the elf file.
- Download the Android *Native Development Kit* (NDK) from the *Android NDK downloads page* and extract its contents.

The following example demonstrates how to compile a sample C program named `program.c` using Clang from the Android NDK:

```
<ndk-path>/toolchains/llvm/prebuilt/linux-x86_64/bin/clang --target=aarch64-linux-
android34 \
    --sysroot=<ndk-path>/toolchains/llvm/prebuilt/linux-x86_64/sysroot -fdebug-info-
for-profiling \
    -gline-tables-only -Wl,--build-id=sha1 -Wl,--no-rosegment program.c -o program
```

About this task

This procedure describes how to do the following:

- Upload the resulting `program` binary object to the FVP
- Generate and convert the execution trace into source-level profiles
- Download and reuse the result to optimize subsequent compiler builds

See *AutoFDO GitHub repository* for more information about the AutoFDO process in Arm.

Procedure

1. Connect to the FVP running instance. For more information, see Using ADB connections to the FVP.
2. Upload the previous resulting `program` binary object to the remote `/data` path location. For more information, see Upload a file to the FVP.
3. Run the following commands from the **terminal_uart_ap** window to change your privilege level to `root`, which is critical for subsequent steps:

   ```
   cd /storage/self
   su
   chmod a+x /data/program
   ```

4. Record the execution trace of the program.
   The `simpleperf` tool in Android records the execution trace of the application. The `cs_etm` event from Simpleperf captures this trace and stores it in a `perf.data` file.

   To use the Simpleperf tool to record the execution trace of the `program` application, enter the following command on the FVP **terminal_uart_ap** window:

   ```
   simpleperf record -e cs-etm ./program
   ```

   For more information on the Simpleperf tool, see the *Android Simpleperf tool documentation*.
5. Convert the execution trace to instruction samples with branch histories.
   Use the Simpleperf tool to convert the execution trace to an instruction profile. Enter the following command on the FVP **terminal_uart_ap** window to decode the execution trace and generate branch histories in text format accepted by AutoFDO:

   ```
   simpleperf inject -i perf.data -o inj.data --output autofdo --binary program
   ```

6. Convert the instruction samples to source-level profiles:
   The *AutoFDO tool* converts the instruction profiles to source profiles for the GCC, Clang, and LLVM compilers. To install AutoFDO, enter the following command:

   ```
   sudo apt-get install autofdo
   ```

   To convert the instruction samples to source-level profiles, you must pull the instruction profile (generated in the previous step and saved as the `inj.data` file), from the FVP to the host machine using the `adb` command. For more information, see Download a file from the FVP.

   To generate source-level profiles for the compiler, the instruction samples that the `simpleperf inject` command produces are passed to the AutoFDO tool. For example to do this for Clang and LLVM, enter the following command on the host machine):

   ```
   create_llvm_prof --binary program --profile inj.data \
       --profiler text --out program.llvmprof --format text
   ```

7. Use the source-level profile with the compiler.

To optimize the next build of the `program` application, provide the profile produced in the preceding steps to the compiler. For Clang, use the `-fprofile-sample-use` compiler option as follows on the host machine:

```
<ndk-path>/toolchains/llvm/prebuilt/linux-x86_64/bin/clang --target=aarch64-
linux-android34 \
      --sysroot=<ndk-path>/toolchains/llvm/prebuilt/linux-x86_64/sysroot -O2 \
      -fprofile-sample-use=program.llvmprof -o program program.c
```

7.2 Using ADB connections to the FVP

Use the *Android Debug Bridge* (ADB) protocol to connect to the FVP and perform the following basic tasks:

- Upload a file
- Download a file
- Execute a command in the ADB shell

7.2.1 Connect to the FVP

To connect to the FVP, follow the procedure in this section.

About this task

If you run more than one FVP on the same host, each instance is assigned a different ADB port. The assigned ADB port is displayed when you start the FVP. Note the assigned ADB port and modify the commands in this section if necessary, replacing the default port `5555` or `<fvp adb port>` with the correct port.

Procedure

1. Run the FVP and wait for the instance to fully boot. This takes a considerable amount of time on Android and also depends on the host hardware specification. Once the Android distribution boots completely, the `RD Lumex DP0` window displays the complete Android home screen.
2. Enter the following commands on a new host terminal session to connect to the FVP using the ADB protocol:

   ```
   adb connect 127.0.0.1:5555
   adb devices
   ```

   For example:

   ```
   # adb connect 127.0.0.1:5555
   * daemon not running; starting now at tcp:5037
   * daemon started successfully
   connected to 127.0.0.1:5555
   # adb devices
   List of devices attached
   127.0.0.1:5555 offline
   ```

Note

If the connection fails, wait and then retry it. Connection failures can occur when the FVP Android instance takes too long to start the required services.

7.2.2 Upload a file to the FVP

To upload a file to the FVP, follow this procedure.

Procedure

1. Establish an ADB connection to the FVP as described in Connect to the FVP.
2. Enter the following command to upload a local file to the remote FVP instance:

```
adb -s <fvp adb port> push <local host location for original file> <remote
 absolute path location to save file>
```

Note

If the ADB connection is lost before running the `adb push` command, repeat the connection steps and reenter the command.

7.2.3 Download a file from the FVP

To download a file from the FVP, follow this procedure.

Procedure

1. Establish an ADB connection to the FVP as described in Connect to the FVP.
2. Enter the following command to download a remote file to your local host system:

```
adb -s <fvp adb port> pull <remote absolute path location for original file>
 <local host location where to save file>
```

Note

If the ADB connection is lost before running the `adb pull` command, repeat the connection steps and reenter the command.

7.2.4 Execute a remote command on the FVP

To execute a remote command, follow this procedure.

Procedure

1. Establish an ADB connection to the FVP as described in Connect to the FVP.

2. Enter the following command:

```
adb -s <fvp adb port> shell <command>
```

For example:

```
adb -s 127.0.0.1:5555 shell ls -la
```

Use the script `build-scripts/unit_test/adb_verify.sh` to test all ADB commands on the Lumex Android distribution.

Note

If the ADB connection is lost before running the `adb shell` command, repeat the connection steps and reenter the command.

7.3 Configuring the rotational scheduler

The rotational scheduler is a vendor module in the Linux kernel that optimizes CPUs on asymmetric platforms. Typically, on an asymmetric platform, tasks that run on big CPUs finish sooner. Little and medium CPUs can become idle when their tasks migrate to the big CPUs.

The rotational scheduler rotates tasks between CPUs to ensure all the tasks finish approximately at the same time and that there is minimum idle time on any CPU. The rotational scheduler starts when one CPU reaches the Rotate state and ends when there are no CPUs in the Rotate state anymore.

There are sysfs interfaces to configure rotating scheduler:

Enable

Enables or disables the rotating scheduler.

Max_latency_us

Keeps track of the amount of work each rotating task has achieved. At any time, if the task the most ahead finishes, all the rotating tasks should finish within the next `max_latency_us`.

Min_residency_us

Tasks are guaranteed a minimum residency time after a rotation. This prevents from having tasks constantly switching on a CPU. `Min_residency_us` is stronger than `max_latency_us`, meaning that `min_residency_us` is strictly respected and `max_latency_us` is a soft target.

To test the rotation scheduler, see Test the rotational scheduler.

7.4 Using a TAP interface

Use the network tap (TAP) interface on the host to connect to the FVP with ADB networking.

7.4.1 Setting up a TAP interface

Use this procedure to set up the TAP interface. Enter all commands on the host system unless otherwise noted.

About this task

This procedure relies on `libvirt` handling the network bridge. Using `libvirt` ensures that the primary network interface continues operating if the network is misconfigured.

Procedure

1. Install `libvirt` on your development host system:

   ```
   sudo apt-get update && sudo apt-get install libvirt-daemon-system libvirt-clients
   ```

   The host system lists a new interface with a name similar to `virbr0` and an IP address of `192.168.122.1`. Verify these values with the command `ifconfig -a` (or `ip a s` for newer distributions). For example:

   ```
   $ ifconfig -a
   virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
      inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
      ether XX:XX:XX:XX:XX:XX  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

   virbr0-nic: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether XX:XX:XX:XX:XX:XX  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
   $
   ```

2. Create the `tap0` interface:

   ```
   sudo ip tuntap add dev tap0 mode tap user $(whoami)
   sudo ifconfig tap0 0.0.0.0 promisc up
   sudo brctl addif virbr0 tap0
   ```

3. Download and install the Android SDK from Android Developer website or, alternatively, install the `adb` tool package as follows:

   ```
   sudo apt-get install adb
   ```

4. Run the FVP model providing the additional parameter `-t "tap0"` to enable the TAP interface:

   ```
   ./run-scripts/tc4/run_model.sh -m <model binary path> -d android -t "tap0" -a
    <true|false>
   ```

   - If AVB is not enabled, the `-a <true|false>` segment can be omitted .

- If `$TC_GPU` isn't defined yet, provide `--gpu <value>` value, using `--gpu <swr|hwr-prebuilt>`

Before proceeding, allow the Android FVP to fully boot to the Android home screen on the **RD Lumex DP0** window.


Note

Booting the Android FVP takes approximately one hour or more, depending on your host system hardware specification.

5. Once the Android FVP boots, the Android instance has an IP address similar to `192.168.122.62`, as shown in the following `ifconfig` output:

```
console:/ $ ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope: Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:80 TX bytes:80

dummy0    Link encap:Ethernet  HWaddr xxxxxx
          inet6 addr: xxxxxx Scope: Link
          UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:3974

eth0      Link encap:Ethernet  HWaddr xxxxxx  Driver smc91x
          inet addr:192.168.122.62  Bcast:192.168.122.255  Mask:255.255.255.0
          inet6 addr: xxxxxx Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7842 TX bytes:13704
          Interrupt:59 Base address:0xd000 DMA chan:ff

console:/ $
```

6. Validate the connection between the host `tap0` interface and the Android FVP by running the following command on the FVP, in the **terminal_uart_ap** window:

```
ping 192.168.122.1
```

Alternatively, you can validate that the FVP can reach a valid internet gateway by pinging, for example, the IP address `8.8.8.8`.

Results

To establish an ADB connection with the IP address and upload or download files, see Using ADB connections to the FVP.

7.4.2 Shutting down a TAP interface

Use this procedure to disable and remove the `tap0` interface, and to revert your host system to its previous state.

Procedure

1. Remove the `tap0` interface from the bridge configuration:
```
sudo brctl delif virbr0 tap0
```
2. Disable the bridge interface:
```
sudo ip link set virbr0 down
```
3. Remove the bridge interface:
```
sudo brctl delbr virbr0
```
4. Remove the `libvirtd` package:
```
sudo apt-get remove libvirt-daemon-system libvirt-clients
```
5. Disable the tap0 interface
```
sudo ip link set tap0 down
```
6. Delete the tap0 interface
```
sudo ip tuntap del dev tap0 mode tap
```
7. Verify that the tap0 interface has been removed
```
ifconfig -a
```

7.5 Updating the firmware

Firmware is updated from linux userspace using a secure world Trusted Services App. To perform a firmware update, a Firmware Image Package (FIP) must be provided. A FIP is generated during the build process and is located at `<TC_OUTPUT>/deploy`.

This feature is supported only with the Buildroot distribution.

This update procedure is compatible with the latest TF-A implementation and aligns with the *Arm® Client Platform Security Architecture Firmware Framework, version 1.0*.

Obtaining and Uploading a FIP to the running Lumex FVP

We recommend rebuilding TF-A after booting Lumex to generate a FIP with a different timestamp, in order to verify that the Firmware Update App has loaded the new FIP. Note that the running Lumex FVP should still be running during rebuilding.

1. Build an FIP with a different timestamp. Run the following commands after booting Lumex:

```
cd build-scripts
./run_docker.sh ./build-tfa.sh all with_reqs
```

2. Upload the FIP to the running FVP Run the following commands:

```
cd <TC_OUTPUT>/deploy
# the following command assumes that the port 8022 is being used as specified in
 the run_model.sh script
scp -P 8022 fip-tc.bin root@localhost:/root/
# password (if required): root
```

3. Run the Firmware Update App. In the `terminal_uart_ap`, run the following commands:

```
cd /root
ts-fwu-app-nwd fip-tc.bin

# Expected output
    active: 0, prev active 0
    num of images 1
```

4. Verify that the FIP has been loaded.
   a. View TF-A logs in `terminal_uart1_ap` to see the timestamps of when the FIP was built. An example of these logs is as follows:

```
NOTICE:  Booting Trusted Firmware
NOTICE:  BL1: v2.12.0(release):v2.9.0-3350-gf54168578
NOTICE:  BL1: Built : 11:22:07, Jul 17 2025
NOTICE:  BL1: Booting BL2
NOTICE:  BL2: v2.12.0(release):v2.9.0-3350-gf54168578
NOTICE:  BL2: Built : 11:22:07, Jul 17 2025
NOTICE:  BL1: Booting BL31
NOTICE:  BL31: v2.12.0(release):v2.9.0-3350-gf54168578
NOTICE:  BL31: Built : 11:22:08, Jul 17 2025
```

   b. After running the Firmware Update App, run the following command in `terminal_uart_ap`:

```
reboot
```

   c. During reboot, check to see that the TF-A logs indicate a different build timestamp:

```
NOTICE:  Booting Trusted Firmware
NOTICE:  BL1: v2.12.0(release):v2.9.0-3350-gf54168578
NOTICE:  BL1: Built : 15:30:47, Jul 18 2025
NOTICE:  BL1: Booting BL2
NOTICE:  BL2: v2.12.0(release):v2.9.0-3350-gf54168578
NOTICE:  BL2: Built : 15:30:47, Jul 18 2025
NOTICE:  BL1: Booting BL31
NOTICE:  BL31: v2.12.0(release):v2.9.0-3350-gf54168578
NOTICE:  BL31: Built : 15:30:47, Jul 18 2025
```

8. System profiling and tracing

You can use the tools, methodologies, and features described in this section to profile the performance and behavior of the system and applications.

8.1 Simpleperf tool

Simpleperf is a native CPU profiling tool for Android. You can use Simpleperf to profile both Android applications and native processes running on Android.

To learn more about Simpleperf, see the *Android Simpleperf tool README*.

The Linux kernel exposes several *Performance Monitoring Unit* (PMU) events to the user space. These PMU events include both CPU and non-CPU PMU events, as well as software and tracepoint events. The kernel exposes these events using the `perf_event_open` system call that Simpleperf uses to collect and process the event data.

The following subsections describe `simpleperf` commands you can use to obtain useful information for troubleshooting and validation. For detailed command information see the *Android Simpleperf tool command reference*.


Note

Most Simpleperf commands require root privileges to retrieve system-wide information. To ensure that you have the required access, run the command `su 0` before you run the Simpleperf example commands.

8.1.1 Simpleperf list command

The `simpleperf list` command lists the supported events.

Syntax

```
console:/ $ simpleperf list
```

Options

The example in this section shows a `simpleperf list` command with no options. For a full list of options, either enter `simpleperf list --help` or see the *Android Simpleperf tool command reference*. For more information, see the sections “Common event numbers” and “**IMPLEMENTATION DEFINED** event numbers” in the *Arm® Architecture Reference Manual for A-profile architecture*.

Example output

```
List of hw-cache events:
  # More cache events are available in `simpleperf list raw`.
  branch-load-misses
```

```
  branch-loads
  dTLB-load-misses
  dTLB-loads
  iTLB-load-misses
  iTLB-loads
  L1-dcache-load-misses
  L1-dcache-loads
  L1-icache-load-misses
  L1-icache-loads
  LLC-load-misses
  LLC-loads

List of coresight etm events:
  cs_etm/autofdo/
  cs-etm        # CoreSight ETM instruction tracing

List of hardware events:
  branch-instructions
  branch-misses
  bus-cycles
  cache-misses
  cache-references
  cpu-cycles
  instructions
  stalled-cycles-backend
  stalled-cycles-frontend

List of pmu events:
  arm_cspmu_0/cycles/
  arm_cspmu_1/cycles/
  arm_cspmu_2/cycles/
  arm_cspmu_3/cycles/
  arm_cspmu_4/cycles/
  arm_cspmu_5/cycles/
  arm_cspmu_6/cycles/
  arm_cspmu_7/cycles/
  arm_dsu_0/bus_access/
  arm_dsu_0/bus_cycles/
  arm_dsu_0/cycles/
  arm_dsu_0/memory_error/
  armv9_c1_nano/br_immed_retired/
  armv9_c1_nano/br_mis_pred/
  armv9_c1_nano/br_retired/
  armv9_c1_nano/br_return_retired/
  armv9_c1_nano/bus_access/
      .
      .
      .
  armv9_c1_pro/br_immed_retired/
  armv9_c1_pro/br_mis_pred/
  armv9_c1_pro/br_retired/
  armv9_c1_pro/br_return_retired/
  armv9_c1_pro/bus_access/
      .
      .
      .
  armv9_c1_ultra/br_immed_retired/
  armv9_c1_ultra/br_mis_pred/
  armv9_c1_ultra/br_mis_pred_retired/
  armv9_c1_ultra/br_pred/
  armv9_c1_ultra/br_retired/
  armv9_c1_ultra/br_return_retired/
  armv9_c1_ultra/bus_access/

List of raw events provided by cpu pmu:
  # Please refer to "PMU common architectural and microarchitectural event numbers"
  # and "ARM recommendations for IMPLEMENTATION DEFINED event numbers" listed in
  # ARMv9 manual for details.
  # A possible link is https://developer.arm.com/documentation/ddi0487.
  raw-ase-fp-addsub-spec (may supported on cpu 0-7)     # Floating-point operation
 speculatively executed, Advanced SIMD add or subtract
```

```
  raw-ase-fp-cvt-spec (may supported on cpu 0-7)        # Floating-point operation
 speculatively executed, Advanced SIMD convert
  raw-ase-fp-div-spec (may supported on cpu 0-7)        # Floating-point operation
 speculatively executed, Advanced SIMD divide
  raw-ase-fp-dot-spec (may supported on cpu 0-7)        # Floating-point operation
 speculatively executed, Advanced SIMD dot-product
  raw-ase-fp-dp-spec (may supported on cpu 0-7)     # Floating-point operation
 speculatively executed, Advanced SIMD double precision
  raw-ase-fp-fma-spec (may supported on cpu 0-7)        # Floating-point operation
 speculatively executed, Advanced SIMD FMA
  raw-ase-fp-hp-spec (may supported on cpu 0-7)     # Floating-point operation
 speculatively executed, Advanced SIMD half precision
  raw-ase-fp-mmla-spec (may supported on cpu 0-7)       # Floating-point operation
 speculatively executed, Advanced SIMD matrix multiply
  raw-ase-fp-mul-spec (may supported on cpu 0-7)        # Floating-point operation
 speculatively executed, Advanced SIMD multiply
  raw-ase-fp-preduce-spec (may supported on cpu 0-7)        # Floating-point
 operation speculatively executed, Advanced SIMD pairwise add step
  raw-ase-fp-recpe-spec (may supported on cpu 0-7)      # Floating-point operation
 speculatively executed, Advanced SIMD reciprocal estimate
  raw-ase-fp-sp-spec (may supported on cpu 0-7)     # Floating-point operation
 speculatively executed, Advanced SIMD single precision
       .
       .
       .
  raw-unaligned-ld-spec (may supported on cpu 0-7)      # Unaligned access, read
  raw-unaligned-ldst-retired (supported on cpu 0-5, may supported on cpu 6-7)
  # Instruction architecturally executed, Condition code check pass, unaligned load
 or store
  raw-unaligned-ldst-spec (may supported on cpu 0-7)        # Unaligned access
  raw-unaligned-st-spec (may supported on cpu 0-7)      # Unaligned access, write
  raw-uop-retired (may supported on cpu 0-7)        # Micro-operation
 architecturally executed
  raw-uop-spec (may supported on cpu 0-7)       # Microarchitectural operation
 speculatively executed
  raw-vfp-spec (may supported on cpu 0-7)       # Operation speculatively executed,
 scalar floating-point

List of software events:
  alignment-faults
  context-switches
  cpu-clock
  cpu-migrations
  emulation-faults
  major-faults
  minor-faults
  page-faults
  task-clock

List of tracepoint events:
  alarmtimer:alarmtimer_cancel
  alarmtimer:alarmtimer_fired
  alarmtimer:alarmtimer_start
  alarmtimer:alarmtimer_suspend
  asoc:snd_soc_bias_level_done
  asoc:snd_soc_bias_level_start
  asoc:snd_soc_dapm_connected
  asoc:snd_soc_dapm_done
  asoc:snd_soc_dapm_path
  asoc:snd_soc_dapm_start
  asoc:snd_soc_dapm_walk_done
  asoc:snd_soc_dapm_widget_event_done
  asoc:snd_soc_dapm_widget_event_start
  asoc:snd_soc_dapm_widget_power
  asoc:snd_soc_jack_irq
  asoc:snd_soc_jack_notify
  asoc:snd_soc_jack_report
  avc:selinux_audited
  binder:binder_alloc_lru_end
  binder:binder_alloc_lru_start
  binder:binder_alloc_page_end
```

```
  binder:binder_alloc_page_start
      .
      .
      .
  xhci-hcd:xhci_setup_device_slot
  xhci-hcd:xhci_stop_device
  xhci-hcd:xhci_urb_dequeue
  xhci-hcd:xhci_urb_enqueue
  xhci-hcd:xhci_urb_giveback
```

8.1.2 Simpleperf stat command

The `simpleperf stat` command returns the event counter values of the profiled processes.

You can use filters to specify the following:

- Which events to use
- Which processes or threads to monitor
- How to monitor
- What print interval to adopt

Syntax

```
simpleperf stat [options] [command [command-args]]
```

Parameters

The command examples use the following options. For a full list of options, either enter `simpleperf stat --help` or see the *Android Simpleperf tool command reference*.

`-a`

Collect system-wide information

`--duration time_in_sec`

Monitor for the specified number of seconds instead of running `[command]`. Specify `time_in_sec` as any positive floating point number.

`-e event1[:modifier1],event2[:modifier2],...`

Select a list of events to count. An event can be either of the following:

- Named events as displayed by `simpleperf list`
- Raw PMU events in `rN` format where `N` is a hex number. For example, `r1b` selects event number `0x1b`.

You can add modifiers to define how to monitor the event. For example:

- Use `u` to monitor user space events only.
- Use `k` to monitor kernel space events only.

`--group event1[:modifier],event2[:modifier2],...`

Similar to the `-e` option, but events specified in the same `--group` option are monitored as a group, and scheduled in and out at the same time.

--interval time_in_ms

Print stat for the specified number of milliseconds. Specify `time_in_ms` as any positive floating point number. Simpleperf prints total values from the starting point. But this can be changed by `--interval-only-values`.

-p

Stat events on existing processes. Processes are searched either by pid or process name regex. Mutually exclusive with the `-a` option.

-t

Stat events on existing threads. Mutually exclusive with the `-a` option.

Example: Get system-wide event counts

This example returns system-wide default event counts, with a duration of 1 second, printing the counts every 50 ms.

```
console:/ # simpleperf stat -a --duration 1 --interval 0.05

Performance counter statistics:

#         count  event_name                # count / runtime
      1,483,234  cpu-cycles                # 0.077020 GHz
              0  stalled-cycles-frontend   # 0.000 /sec
              0  stalled-cycles-backend    # 0.000 /sec
      1,297,390  instructions              # 1.143245 cycles per instruction
        192,137  branch-instructions       # 10.411 M/sec
              0  branch-misses             # 0.000000% miss rate
  17.551168(ms)  task-clock                # 16.392911 cpus used
              2  context-switches          # 119.125 /sec
             14  page-faults               # 870.827 /sec

Total test time: 0.001071 seconds.
Performance counter statistics:

#         count  event_name                # count / runtime
      2,289,588  cpu-cycles                # 0.067113 GHz
              0  stalled-cycles-frontend   # 0.000 /sec
              0  stalled-cycles-backend    # 0.000 /sec
      2,087,677  instructions              # 1.096716 cycles per instruction
        334,399  branch-instructions       # 10.137 M/sec
              0  branch-misses             # 0.000000% miss rate
  32.023176(ms)  task-clock                # 10.628477 cpus used
              4  context-switches          # 127.891 /sec
             15  page-faults               # 490.151 /sec

Total test time: 0.003013 seconds.
.
.
.
```

Example: Get event counts for a specific process within a duration

This example gets the default event counts for the process `system_server` with a duration of 50 ms.

```
console:/ # ps -A | grep system_server
system         477   318   19313020 338596 do_epoll_wait       0 S system_server
console:/ #
console:/ # simpleperf stat -p 477 --duration 0.05
```

```
    Performance counter statistics:

    #        count  event_name                # count / runtime
                 0  cpu-cycles                #
                 0  stalled-cycles-frontend   #
                 0  stalled-cycles-backend    #
                 0  instructions              #
                 0  branch-instructions       #
                 0  branch-misses             #
      0.000000(ms)  task-clock                # 0.000000 cpus used
                 0  context-switches          #
                 0  page-faults               #

    Total test time: 0.050069 seconds.
    console:/sdcard #
```

This example returns counts of 0 because the PMU events are not implemented on the FVP.

Example: Get specific events for a particular process

This example shows how to get the events for the process “system_server” process with PID 477.

```
    console:/ # simpleperf stat -e cpu-cycles -p 477 --duration 0.05
    Performance counter statistics:

    #     count  event_name   # count / runtime
      6,351,580  cpu-cycles   # 0.099210 GHz

    Total test time: 0.050210 seconds.
    console:/sdcard #

    # Additional examples:
    console:/ # simpleperf stat -e cache-references,cache-misses -p 477 --duration
0.05
    console:/ # simpleperf stat -e cache-references,cache-misses ls
```

In the same way you can filter events for a particular process using `-p <PID>` option, you can also filter events for specific threads using `-t <TID>` option.

Example: Get non-CPU PMU events

This example shows how to list non-CPU PMU events. You cannot list non-CPU PMU events per process because Simpleperf cannot attach events to a process.

```
    console:/ # simpleperf stat -a -e arm_dsu_0/cycles/ -- sleep 0.01
    Performance counter statistics:

    #                     count  event_name          # count / runtime
      9,223,372,036,854,775,809  arm_dsu_0/cycles/   # 433003296234.548 G/sec

    Total test time: 0.021216 seconds.
    console:/sdcard #
```

Example: Collect event counters using event-groups

This example shows how to use event groups.

```
    console:/ # simpleperf stat --group cpu-cycles,instructions -- ls

    acct         debug_ramdisk          lost+found   second_stage_resources
    apex         dev                    mnt          storage
```

```
bin           etc                      odm           sys
bugreports    fstab.total_compute      odm_dlkm      system
cache         init                     oem           system_dlkm
config        init.common.rc           postinstall   system_ext
d             init.environ.rc          proc          vendor
data          init.total_compute.rc    product       vendor_dlkm
data_mirror   linkerconfig             sdcard
Performance counter statistics:

#      count  event_name     # count / runtime
  23,287,967  cpu-cycles     # 1.507257 GHz
  23,287,967  instructions   # 1.000000 cycles per instruction

Total test time: 0.018505 seconds.
console:/sdcard #
```

8.1.3 Simpleperf record command

The `simpleperf record` command dumps samples of the profiled processes.

Syntax

```
simpleperf record [options] [--] [command [command-args]]
```

Options

The command examples use the following options. For a full list of options, either enter `simpleperf record --help` or see the *Android Simpleperf tool command reference*.

`-p pid_or_process_name_regex1,pid_or_process_name_regex2,...`

Record events on existing processes. Processes are searched either by pid or process name regex.

`--duration time_in_sec`

Monitor for `time_in_sec` seconds instead of running `[command]`. Specify `time_in_sec` as any positive floating point number.

`-f freq`

Set event sample frequency. Records a maximum of the specified `freq` samples every second. For non-tracepoint events, the default option is `-f 4000`. A `-f/-c` option affects all subsequent event types until the next `-f/-c` option. For example, for `-f 1000 cpu-cycles -c 1 -e sched:sched_switch`, cpu-cycles has the sample frequency 1000, and the sched:sched_switch event has the sample period 1.

`-c count`

Set event sample period. It means recording 1 sample when `count` events happen. For tracepoint events, the default option is `-c 1`.

Example: Basic usage

The following example shows a basic `simpleperf record` command that dumps samples from the `ls` command.

```
console:/sdcard # simpleperf record ls
Alarms      DCIM       Movies         Pictures    Ringtones
```

```
    Android     Documents  Music          Podcasts    TemporaryFile-t57Mnj
    Audiobooks  Download   Notifications  Recordings  perf.data
    simpleperf I cmd_record.cpp:798] Recorded for 0.0108992 seconds. Start post
 processing.
    simpleperf I cmd_record.cpp:891] Samples recorded: 37. Samples lost: 0.
    console:/sdcard #
```

Additional examples

This section contains more advanced `simpleperf record` command examples.

To record an individual process for a specific duration, enter the following command:

```
simpleperf record -p <PID> --duration time_in_sec
```

To record set of processes for a specific duration, enter the following command:

```
simpleperf record -p <PID1>,<PID2> --duration time_in_sec
```

To spawn a workload as a child process and record it, enter the following command:

```
simpleperf record <WORKLOAD APPLICATION>
```

To collect 1,000 records every second, enter the following command:

```
simpleperf record -f 1000 -p <PID> --duration time_in_sec
```

To collect a record when 1,000 events are reached, enter the following command:

```
simpleperf record -c 1000 -p <PID> --duration time_in_sec
```

8.1.4 Simpleperf report command

The `simpleperf report` command generates a report of the profiling data generated by the `simpleperf record` command.

The following example shows how to generate a report based on the previously-entered `simpleperf record ls` command:

```
console:/sdcard # simpleperf report
Cmdline: /system/bin/simpleperf record ls
Arch: arm64
Event: cpu-cycles (type 0, config 0)
Samples: 34
Event count: 22952710

    Overhead  Command  Pid   Tid   Shared Object                          Symbol
    36.66%    ls       6446  6446  /system/lib64/libcrypto.so
 sha256_block_data_order_nohw
```

```
    7.96%     ls       6446  6446  /apex/com.android.runtime/bin/linker64
 [linker]soinfo::lookup_version_info(VersionTracker const&, unsigned int, char
 const*, version_info const**)
    7.23%     ls       6446  6446  [kernel.kallsyms]                        call_rcu
    6.83%     ls       6446  6446  [kernel.kallsyms]                        el0_svc
    5.46%     ls       6446  6446  [kernel.kallsyms]
 mas_destroy
    5.10%     ls       6446  6446  /apex/com.android.runtime/bin/linker64
  [linker]Config::read_binary_config(char const*, char const*, bool, bool,
 Config const**, std::__1::basic_string<char, std::__1::char_traits<char>,
 std::__1::allocator<char>>*)
    5.08%     ls       6446  6446  /apex/com.android.runtime/bin/linker64
 [linker]BionicSmallObjectAllocator::alloc()
    4.76%     ls       6446  6446  [kernel.kallsyms]
 mas_empty_area_rev
    4.55%     ls       6446  6446  /apex/com.android.runtime/bin/linker64
 [linker]mprotect
    4.38%     ls       6446  6446  [kernel.kallsyms]
 down_write
    3.84%     ls       6446  6446  /apex/com.android.runtime/bin/linker64
 [linker]bool plain_relocate_impl<(RelocMode)0>(Relocator&, elf64_rela*, unsigned
 long) (.__uniq.153370809355997480299804515629147722701)
    3.30%     ls       6446  6446  [kernel.kallsyms]
 folio_remove_rmap_ptes
    3.25%     ls       6446  6446  [kernel.kallsyms]
 mas_push_data
    1.51%     ls       6446  6446  /apex/com.android.runtime/bin/linker64
 [linker]page_size()
    0.10%     ls       6446  6446  [kernel.kallsyms]
 __kasan_slab_alloc
    0.01%     ls       6446  6446  [kernel.kallsyms]
 mas_wr_walk
    0.00%     ls       6446  6446  [kernel.kallsyms]
 ___rmqueue_pcplist
    0.00%     ls       6446  6446  [kernel.kallsyms]
 __kasan_unpoison_pages
    0.00%     ls       6446  6446  [kernel.kallsyms]
 _get_random_bytes.llvm.8911717040554631468
    0.00%     ls       6446  6446  [kernel.kallsyms]
 flush_signal_handlers
    0.00%     ls       6446  6446  [kernel.kallsyms]
 update_sctlr_el1

console:/sdcard #
```

8.2 Perf profiler tool

Perf is a profiler tool for Linux-based systems that abstracts CPU hardware differences to measure Linux performance. The perf tool has a simple command-line interface.

The Linux kernel exposes several *Performance Monitoring Unit* (PMU) events to the user space. These PMU events include both CPU and non-CPU PMU events, as well as software and tracepoint events. The kernel exposes these events using the `perf_event_open` system call that perf uses to collect and process the event data.

For more information on the tool, see the *perf wiki*.

8.2.1 Perf list command

The `perf list` command displays the event types you can select in the various perf commands with the `-e` option.

Syntax

```
# perf list
```

Options

The example in this section shows a `perf list` command with no options. For a full list of options, either enter `perf list -h` or see the *perf documentation*.

Example output

```
    # perf list
    List of pre-defined events (to be used in -e or -M):

      branch-instructions OR branches                    [Hardware event]
      branch-misses                                      [Hardware event]
      bus-cycles                                         [Hardware event]
      cache-misses                                       [Hardware event]
      cache-references                                   [Hardware event]
      cpu-cycles OR cycles                               [Hardware event]
      instructions                                       [Hardware event]
      stalled-cycles-backend OR idle-cycles-backend      [Hardware event]
      stalled-cycles-frontend OR idle-cycles-frontend    [Hardware event]

      alignment-faults                                   [Software event]
      bpf-output                                         [Software event]
      cgroup-switches                                    [Software event]
      context-switches OR cs                             [Software event]
      cpu-clock                                          [Software event]
      cpu-migrations OR migrations                       [Software event]
      dummy                                              [Software event]
      emulation-faults                                   [Software event]
      major-faults                                       [Software event]
      minor-faults                                       [Software event]
      page-faults OR faults                              [Software event]
      task-clock                                         [Software event]

      duration_time                                      [Tool event]
      user_time                                          [Tool event]
      system_time                                        [Tool event]

armv9_c1_pro:
      L1-dcache-loads OR armv9_c1_pro/L1-dcache-loads/
      L1-dcache-load-misses OR armv9_c1_pro/L1-dcache-load-misses/
      L1-icache-loads OR armv9_c1_pro/L1-icache-loads/
      L1-icache-load-misses OR armv9_c1_pro/L1-icache-load-misses/
      LLC-loads OR armv9_c1_pro/LLC-loads/
      LLC-load-misses OR armv9_c1_pro/LLC-load-misses/
      dTLB-loads OR armv9_c1_pro/dTLB-loads/
      dTLB-load-misses OR armv9_c1_pro/dTLB-load-misses/
      iTLB-loads OR armv9_c1_pro/iTLB-loads/
      iTLB-load-misses OR armv9_c1_pro/iTLB-load-misses/
      branch-load-misses OR armv9_c1_pro/branch-load-misses/

    armv9_c1_nano:
      L1-dcache-loads OR armv9_c1_nano/L1-dcache-loads/
      L1-dcache-load-misses OR armv9_c1_nano/L1-dcache-load-misses/
      L1-icache-loads OR armv9_c1_nano/L1-icache-loads/
      L1-icache-load-misses OR armv9_c1_nano/L1-icache-load-misses/
      LLC-loads OR armv9_c1_nano/LLC-loads/
```

```
      LLC-load-misses OR armv9_c1_nano/LLC-load-misses/
      dTLB-loads OR armv9_c1_nano/dTLB-loads/
      dTLB-load-misses OR armv9_c1_nano/dTLB-load-misses/
      iTLB-loads OR armv9_c1_nano/iTLB-loads/
      iTLB-load-misses OR armv9_c1_nano/iTLB-load-misses/
      branch-load-misses OR armv9_c1_nano/branch-load-misses/

    armv9_c1_nano:
      L1-dcache-loads OR armv9_c1_nano/L1-dcache-loads/
      L1-dcache-load-misses OR armv9_c1_nano/L1-dcache-load-misses/
      L1-icache-loads OR armv9_c1_nano/L1-icache-loads/
      L1-icache-load-misses OR armv9_c1_nano/L1-icache-load-misses/
      LLC-loads OR armv9_c1_nano/LLC-loads/
      LLC-load-misses OR armv9_c1_nano/LLC-load-misses/
      dTLB-loads OR armv9_c1_nano/dTLB-loads/
      dTLB-load-misses OR armv9_c1_nano/dTLB-load-misses/
      iTLB-loads OR armv9_c1_nano/iTLB-loads/
      iTLB-load-misses OR armv9_c1_nano/iTLB-load-misses/
      branch-loads OR armv9_c1_nano/branch-loads/
      branch-load-misses OR armv9_c1_nano/branch-load-misses/
      br_immed_retired OR armv9_c1_pro/br_immed_retired/  [Kernel PMU event]
      br_mis_pred OR armv9_c1_pro/br_mis_pred/            [Kernel PMU event]
      br_retired OR armv9_c1_pro/br_retired/              [Kernel PMU event]
      br_return_retired OR armv9_c1_pro/br_return_retired/[Kernel PMU event]

      arm_cspmu_0/cycles/                                 [Kernel PMU event]
      arm_cspmu_1/cycles/                                 [Kernel PMU event]
      arm_cspmu_2/cycles/                                 [Kernel PMU event]
      arm_cspmu_3/cycles/                                 [Kernel PMU event]
      arm_cspmu_4/cycles/                                 [Kernel PMU event]
      arm_cspmu_5/cycles/                                 [Kernel PMU event]
      arm_cspmu_6/cycles/                                 [Kernel PMU event]
      arm_cspmu_7/cycles/                                 [Kernel PMU event]
      arm_dsu_0/bus_access/                               [Kernel PMU event]
      arm_dsu_0/bus_cycles/                               [Kernel PMU event]
      arm_dsu_0/cycles/                                   [Kernel PMU event]
      arm_dsu_0/memory_error/                             [Kernel PMU event]
      arm_spe_0//                                         [Kernel PMU event]
      arm_spe_1//                                         [Kernel PMU event]
      cs_etm//                                            [Kernel PMU event]
      cs_etm/autofdo/                                     [Kernel PMU event]
      rNNN                                                [Raw hardware event
 descriptor]
      mem:<addr>[/len][:access]                           [Hardware breakpoint]
      alarmtimer:alarmtimer_cancel                        [Tracepoint event]
      alarmtimer:alarmtimer_fired                         [Tracepoint event]
      alarmtimer:alarmtimer_start                         [Tracepoint event]
      alarmtimer:alarmtimer_suspend                       [Tracepoint event]
      xhci-hcd:xhci_urb_dequeue                           [Tracepoint event]
      xhci-hcd:xhci_urb_enqueue                           [Tracepoint event]
      xhci-hcd:xhci_urb_giveback                          [Tracepoint event]
```

The `perf list` command displays unformatted output when running on the FVP. To make the output more readable, redirect it to a file and list the contents of the file as follows.

```
# perf list > perf_list.txt
# cat perf_list.txt
```

8.2.2 Perf Stat command

The `perf stat` command executes the specified command, keeps a running count of the hardware and software events during the command execution, and generates statistics for these counts. If

you do not specify any events, the `perf stat` command counts a set of common hardware and software events.

Syntax

```
perf stat [-e <EVENT> | --event=EVENT] [-a] \-- <command> [<options>]
```

Options

For a full list of options, either enter `perf stat -h` or see the *perf documentation*.

Operation

Lumex defines per-microarchitecture PMU instances. As a result, the `perf list` command displays the Kernel CPU PMU events for each CPU micro-architecture. For example:

```
.
.
.
      cpu_cycles OR armv9_c1_pro/cpu_cycles/              [Kernel PMU event]
      cpu_cycles OR armv9_c1_nano/cpu_cycles/             [Kernel PMU event]
      cpu_cycles OR armv9_c1_nano/cpu_cycles/             [Kernel PMU event]
.
.
.
```

For Linux kernel version 6.1 and for situations where the `perf` command executes in a task-bound way (`cpu==-1`), the event opens on an arbitrary CPU PMU and only counts on a subset of CPUs. For example, the event might open on a “big” CPU PMU and only count while the task is running on “big” CPUs, but not while the task is running on “little” CPUs. The following example shows one such situation. The cycles are not counted for the command `ls`, because the command executes on CPUs whose PMUs are not selected by `perf` to open the events:

```
# perf stat -e cycles -- ls
arm-ffa-tee.ko
build_env.cfg

 Performance counter stats for 'ls':

    <not
 counted>      cycles                                                        (0.00%)

    0.000509460 seconds time elapsed

    0.000044000 seconds user
    0.000000000 seconds sys

#
```

To mitigate this behavior and ensure that you always retrieve meaningful data, use the `perf stat` command in either of the following ways:

- Provide the `perf` command with the individual CPU PMU events to count:

```
  # perf stat -e armv9_c1_nano/cpu_cycles/,armv9_c!_pro/
cpu_cycles/,armv9_c1_ultra/cpu_cycles/ -- ls
        arm-ffa-tee.ko
```

```
        build_env.cfg

          Performance counter stats for 'ls':

              <not counted>      armv9_c1_pro/
cycles/                                                       (0.00%)
              <not counted>      armv9_c1_nano/
cycles/                                                      (0.00%)
                    1573935      armv9_c1_nano/cycles/

                0.000853528 seconds time elapsed

                0.000983000 seconds user
                0.000000000 seconds sys
  #
```

- Provide the `perf` command with a CPU mask so the event opens on all CPU PMUs:

```
# perf stat -C 0-7 -e instructions,cycles -- ls
arm-ffa-tee.ko
build_env.cfg

          WARNING: A requested CPU in '0-7' is not supported by PMU
 'armv9_c1_pro'  (CPUs 2-5) for event 'instructions'
          WARNING: A requested CPU in '0-7' is not supported by PMU
 'armv9_c1_nano' (CPUs 0-1) for event 'instructions'
          WARNING: A requested CPU in '0-7' is not supported by PMU
 'armv9_c1_nano' (CPUs 6-7) for event 'instructions'
          WARNING: A requested CPU in '0-7' is not supported by PMU
 'armv9_c1_pro'  (CPUs 2-5) for event 'cycles'
          WARNING: A requested CPU in '0-7' is not supported by PMU
 'armv9_c1_nano' (CPUs 0-1) for event 'cycles'
          WARNING: A requested CPU in '0-7' is not supported by PMU
 'armv9_c1_nano' (CPUs 6-7) for event 'cycles'

 Performance counter stats for 'CPU(s) 0-7':

           154412      armv9_c1_pro/instructions/        #    1.01  insn per
 cycle
            57256      armv9_c1_nano/instructions/       #    1.01  insn per
 cycle
          1969132      armv9_c1_nano/instructions/       #    1.00  insn per
 cycle
           153600      armv9_c1_pro/cycles/
            56742      armv9_c1_nano/cycles/
          1968510      armv9_c1_nano/cycles/

                0.001367792 seconds time elapsed

#
```

As the previous example shows, the instructions are not broken down to specific PMU type (CPU type). This leads to ambiguity in the result because instructions and cycles on different CPUs have different performance meaning. Version 6.6 and later of the Linux kernel do not exhibit this behavior when running the `perf` command with the default event names without providing the CPU mask. A possible solution is to compile `perf` from newer source code and copy the resulting binary into the rootfs before booting the image. Alternatively use the `scp` command to upload the binary to a booted system.

The following table shows `perf stat` command examples that use the `-C 0-7` argument to work around the previous issue (Lumex FVP has 8 CPUs):

Table 8-1: `perf stat` command examples

Statistic	Command
Single event	`perf stat -C 0-7 -e <EVENT> -- <WORKLOAD>`
Multiple events	`perf stat -C 0-7 -e <EVENT1>,<EVENT2>,...,<EVENT-N> -- <WORKLOAD>`
Event group	`perf stat -C 0-7 -e '{<EVENT1>,<EVENT2>,...,<EVENT-N>}' -- <WORKLOAD>`
Existing process. The `sleep` option runs perf for the specified duration	`perf stat -C 0-7 -e <EVENT> -p <PID> -- sleep 1`

DSU and MCN PMU drivers do not support all possible events by name. For cases where data for a particular event is not visible, use `perf stat` with a raw event ID. The following examples show how to read the non-CPU PMU event counters. The values `0xa2` and `0x182` were obtained from the PMU sections of the component TRM documentation, *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual* .

Table 8-2: `perf stat` non-CPU command examples

Statistic	Command
DSU events	`perf stat -e arm_dsu_0/cycles/,arm_dsu_0/memory_error/ -- sleep 0.01`
DSU cache read refills	`perf stat -e arm_dsu_0/event=0xa2/ -- sleep 0.01`
MCN write requests on the memory controller interface	`perf stat -e arm_cspmu_0/event=0x182/ -- sleep 0.01`

8.2.3 Perf record command

The `perf record` command collects sampling data for one or more events, and saves the data in a `perf.data` file. Once created, you can analyze the data in the `perf.data` file using the `perf report` or `perf annotate` commands. By default, the `perf record` command uses cycles as a default event.

Syntax

```
perf record' [-e <EVENT> | --event=EVENT] [-a] \-- <command> [<options>]
```

Options

To modify the sampling period while running `perf record`, do one of the following:

- Use the `-F` option (frequency) to specify the average rate of samples per second
- Use the `-c` option (count) to enforce sampling at the specified event period

For a full list of options, either enter `perf record -h` or see the *perf documentation*.

Examples

The following table shows how to use the `perf record` command.

Table 8-3: `perf record` command examples

Statistic	Command
Event cycles at the default frequency	`perf record -C 0-7 <WORKLOAD>`
Event instructions at 1000 samples per second	`perf record -C 0-7 -e instructions -F 1000 <WORKLOAD>`
Event instructions at every 2000 occurrences of event	`perf record -C 0-7 -e instructions -c 2000 <WORKLOAD>`

8.2.4 Perf and the Arm SPE extension

The *Statistical Profiling Extension* (SPE) feature provides a hardware-assisted CPU operation profiling mechanism that provides the accurate attribution of latencies and events down to individual instructions.

For a discussion of the difference between PMU and SPE events, see the knowledge article *What is the Difference between Arm Statistical Profiling Extension and Performance Monitor Unit Events?*. For a methodology for workload characterization and root cause analysis using the SPE, see the *Arm Statistical Profiling Extension: Performance Analysis Methodology White Paper*.

The basic `perf record` command with the SPE option has the following syntax:

```
perf record -e arm_spe_<spe_instance>/<CONFIG PARAMETERS>/ -- taskset -c <cpu_list>
 <WORKLOAD>
```

Lumex supports SPE only on Mid and Big CPUs and not on small CPUs, there are 2 SPE instances, `arm_spe_0` for Mid CPUs (CPUs 2-5) and `arm_spe_1` for big CPUs (CPUs 6-7). When workload needs to be analyzed using SPE, it should be bound to CPUs which have the SPE capability using taskset. So on Lumex platform workloads should be bound to CPUs 2-5 when using `arm_spe_0` and workloads should be bound to CPUs 6-7 when using `arm_spe_1`. `min_latency=0` config parameter is mandatory to provide with any perf-spe command.

The following listing illustrates how to record SPE samples on Mid CPUs with `arm_spe_0`:

```
# perf record -e arm_spe_0/min_latency=0/ -- taskset -c 2-5 ls
arm-tstee.ko   build_env.cfg  perf.data
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.124 MB perf.data ]
#
```

To analyze the previously-created `perf.data` file, enter the following `perf report` command:

```
# perf report
  Warning:
  Please install libunwind or libdw development packages during the perf build.
  Only instruction-based sampling period is currently supported by Arm SPE.
  # To display the perf.data header info, please use --header/--header-only options.
  #
  #
  # Total Lost Samples: 0
  #
  # Samples: 601  of event 'l1d-access'
  # Event count (approx.): 601
  #
```

```
# Children  Self   Command  Shared Object      Symbol
# ........  .....  .......  .................  ........................
#
   3.16%     3.16%  ls       [kernel.kallsyms]  [k] unmap_page_range
   2.33%     2.33%  ls       [kernel.kallsyms]  [k] next_uptodate_folio
   2.00%     2.00%  ls       [kernel.kallsyms]  [k] filemap_map_pages
   1.83%     1.83%  ls       [kernel.kallsyms]  [k] set_pte_range
   1.83%     1.83%  taskset  [kernel.kallsyms]  [k] unmap_page_range
   1.66%     1.66%  ls       [kernel.kallsyms]  [k] folio_add_file_rmap_ptes
   1.33%     1.33%  ls       [kernel.kallsyms]  [k] bsearch

   .
   .
   .
```

The previous example specified only the required `min_latency=0` configuration paramter. However, specifying other configuration parameters is sometimes useful to filter profiling information. For example, use the configuration parameter `event_filter=2` to discard all samples which do not have retired instructions events:

```
# perf record -e arm_spe_0/min_latency=0,event_filter=2/ -- taskset -c 2 ls
arm-tstee.ko   build_env.cfg  perf.data
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.124 MB perf.data ]
#
```

For detailed information regarding the configuration parameters, see *Support for Arm Statistical Profiling Extension within Perf tools*. For kernel configuration and prerequisites for enabling Arm SPE, see the *perf wiki*.

8.3 Perfetto profiling, tracing, and analysis tool

Perfetto is an open-source software stack for performance instrumentation and trace analysis.

Perfetto includes the following:

- Services and libraries for recording system-level and app-level traces
- Native and Java heap profiling
- A library for analyzing traces using SQL
- A web-based UI that allows to visualize and explore the collected traces

Perfetto uses `ftrace`, `atrace`, `/proc/{stat,vmstat,pid}/*` and `perf_event` as data sources to collect system-level traces. A data source is tracing data that is exposed by a producer. The producer offers the ability to contribute to the trace, advertizing this ability with one or more data sources. A consumer controls the tracing service, provides to the tracing service the trace configuration, and reads back the trace buffers. The tracing service is a long-lived entity, for example a system daemon on Linux or Android. The tracing service handles the tracing sessions, routes trace configuration from consumer to producers, and manages trace buffers.

The data source defines its own schema using Protocol Buffers, consisting of data source trace config (the kind of input config it expects from the consumer) and trace packets (the kind of data it outputs into the trace).

The following are examples of data sources advertized by different producers to collect system-level traces:

- `linux.process_stats`
- `linux.ftrace`
- `linux.sys_stats`
- `linux.perf`

8.3.1 Recording and visualizing traces with Perfetto

You can use Perfetto to record traces, using either the Perfetto web interface or the command-line interface.

The following example shows how to use the Perfetto command-line interface to collect traces:

```
# the following commands are intended to be run on the host PC;
# only applicable for the following command, the current path is assumed to be
 <TC_WORKSPACE>
export PATH="$(pwd)/src/android/out/host/linux-x86/bin:${PATH}"
adb connect localhost:<PORT>
adb devices
adb -s localhost:<PORT> push config.txt /data/local/tmp/config.txt
adb -s localhost:<PORT> shell perfetto -o /data/misc/perfetto-traces/
trace_file.perfetto-trace --txt -c /data/local/tmp/config.txt
adb -s localhost:<PORT> pull /data/misc/perfetto-traces/trace_file.perfetto-trace ./
```


Note

- Do not use the option `-s localhost:<PORT>` if there is only one ADB instance available for debug to the host.
- The default ADB port is `5555`, however in cases where more than one ADB instance is available for debug, the port may be different. See the output of the `adb devices` command or to the FVP model startup log information to determine the correct port. For more information about troubleshooting ADB connections, see Using ADB connections to the FVP.
- The `config.txt` file contains the Perfetto trace config. For examples, see Trace configuration examples.

Once you collect the data in the Perfetto trace file and download it to the host, you can load it to the *Perfetto web UI* using the `Open trace file` menu item. For more information on Perfetto, see the *Perfetto developer docs*.

8.3.2 Trace configuration examples

The following trace configuration examples show how to control the tracing service and influence the sampled data on the Lumex platform. The trace configuration examples also include a visualization of the respective captured trace data, generated using the Perfetto web interface.

For additional examples of data source trace configurations for supported data sources, see the:

- *Perfetto developer docs*, in the "Data sources" section
- `test/configs/` directory in Perfetto source code

In the Perfetto source code, you can also find full lists of the following:

- Supported `ftrace` events in the file `protos/perfetto/trace/ftrace/ftrace_event.proto`
- Supported `meminfo` and `vmstat` counters in the file `protos/perfetto/common/sys_stats_counters.proto`

8.3.2.1 `ftrace` example

This trace configuration file collects `ftrace` scheduling events, process statistics, and system statistic counters every 1000 ms.

```
buffers {
  size_kb: 16384
  fill_policy: RING_BUFFER
}

buffers {
  size_kb: 16384
  fill_policy: RING_BUFFER
}

data_sources {
  config {
    name: "linux.ftrace"
    target_buffer: 0
    ftrace_config {
      # Scheduling information and process tracking. Useful for:
      # - what is happening on each CPU at each moment
      # - why a thread was de-scheduled
      # - parent/child relationships between processes and threads.
      ftrace_events: "sched/sched_switch"
      ftrace_events: "power/suspend_resume"
      ftrace_events: "sched/sched_process_exit"
      ftrace_events: "sched/sched_process_free"
      ftrace_events: "task/task_newtask"
      ftrace_events: "task/task_rename"

      # Wakeup info. Allows to compute how long a task was
      # blocked due to CPU contention.
      ftrace_events: "sched/sched_wakeup"

      # os.Trace markers:
      ftrace_events: "ftrace/print"
      # RSS and ION buffer events:
      ftrace_events: "mm_event/mm_event_record"
      ftrace_events: "kmem/rss_stat"
      ftrace_events: "kmem/ion_heap_grow"
      ftrace_events: "kmem/ion_heap_shrink"
```

```
    }
  }
}

data_sources {
  config {
    name: "linux.sys_stats"
   target_buffer: 1
   sys_stats_config {
     meminfo_period_ms: 100
     meminfo_counters: MEMINFO_MEM_AVAILABLE
     meminfo_counters: MEMINFO_BUFFERS
     meminfo_counters: MEMINFO_CACHED
     meminfo_counters: MEMINFO_SWAP_CACHED
     meminfo_counters: MEMINFO_ACTIVE
     meminfo_counters: MEMINFO_INACTIVE
     meminfo_counters: MEMINFO_ACTIVE_ANON
     meminfo_counters: MEMINFO_INACTIVE_ANON
     meminfo_counters: MEMINFO_ACTIVE_FILE
     meminfo_counters: MEMINFO_INACTIVE_FILE
     meminfo_counters: MEMINFO_UNEVICTABLE

     vmstat_period_ms: 100
     vmstat_counters: VMSTAT_NR_FREE_PAGES
     vmstat_counters: VMSTAT_NR_ALLOC_BATCH
     vmstat_counters: VMSTAT_NR_INACTIVE_ANON
     vmstat_counters: VMSTAT_NR_VMSCAN_WRITE
     vmstat_counters: VMSTAT_NR_VMSCAN_IMMEDIATE_RECLAIM
     vmstat_counters: VMSTAT_NR_WRITEBACK_TEMP

     stat_period_ms: 100
     stat_counters: STAT_CPU_TIMES
     stat_counters: STAT_IRQ_COUNTS
     stat_counters: STAT_FORK_COUNT
    }
  }
}

data_sources: {
    config {
        name: "linux.process_stats"
        target_buffer: 0
        process_stats_config {
            scan_all_processes_on_start: true
            proc_stats_poll_ms: 1000
        }
    }
}

duration_ms: 1000
```

8.3.2.2 `cpu_cycles` example

This trace configuration file collects `cpu_cycles` and instruction CPU PMU counters on all CPUs.

```
    buffers {
      size_kb: 10240
      fill_policy: RING_BUFFER
    }

    data_sources {
      config {
        name: "linux.perf"
        target_buffer: 0
        perf_event_config {
          all_cpus: true
```

```
      timebase {
        frequency: 99
        counter: HW_CPU_CYCLES
        timestamp_clock: PERF_CLOCK_MONOTONIC
      }
    }
  }
}

data_sources {
  config {
    name: "linux.perf"
    target_buffer: 0
    perf_event_config {
      all_cpus: true
      timebase {
        frequency: 99
        counter: HW_INSTRUCTIONS
        timestamp_clock: PERF_CLOCK_MONOTONIC
      }
    }
  }
}

duration_ms: 1000
```

8.3.2.3 Stack sampling example

This trace configuration file calls stack sampling of processes.

```
buffers {
  size_kb: 10240
  fill_policy: RING_BUFFER
}

data_sources {
  config {
    name: "linux.perf"
    target_buffer: 0
    perf_event_config {
      timebase {
        frequency: 99
        timestamp_clock: PERF_CLOCK_MONOTONIC
      }
      callstack_sampling {
        kernel_frames: true
      }
    }
  }
}

duration_ms: 1000
```

9. Programmer's model

The programmer's model describes the Lumex memory maps, interrupt maps, and register definitions.

Rest of System describes the RoS and its memory and interrupt maps.

9.1 Memory maps

Lumex Reference Software has three main memory maps that show how the subsystem memory is structured.

The three main memory maps are:

- Application Processor (AP) system memory map
- System Control Processor (SCP) memory map
- Runtime Security Engine (RSE) memory map

The AP, SCP, and RSE have their own memory maps which are different from each other. The SCP and RSE address spaces are private to these components and cannot be accessed by the AP.

The System address space that the AP operates in can also be accessed by RSE and SCP.

Since SCP and RSE operate in a 32-bit address space, a translation is required for any accesses from SCP and RSE into the System address space which is larger (>32-bit). The Address Translation Unit (ATU) is therefore instantiated in both M-class subsystems. It allows for mapping of certain regions (indicated in the memory map) of M-class addresses to System addresses.

For the required configuration for the ATU, see the *Arm® Corstone™ Reference Systems Architecture Specification Ma1*.

9.1.1 Application Processor memory map

This section details the memory map for the Application Processor (AP) system. The address map is implemented by the SI L1 NoC and is seen by all initiators as a unified view of the system. Access restrictions hide security sensitive regions from individual agents where necessary.

The following tables provide the mapping of address regions to targets and detail access restrictions where applicable.

The System memory map is visible to the following:

- Application Processors (APs)
- System Control Processor (SCP) (through an ATU)
- RSE (through an ATU)

- Embedded Trace Router
- Application Processor Debug Access Port (DAP)

The memory map shows the overall security attributes associated with each area of memory.

Always Secure access

A region that is only accessible to Secure transactions. Any Non-Secure access targeting these, results in a DECERR decode error response.

Secure and Non-Secure access

A region that is accessible to both Secure and Non-Secure transactions.

Programmable access security

Programmable access security is also known as Securable. It is a region that is defined to be independently software-configurable, and can be changed between the following states by trusted software:

- Always Secure access
- Secure and Non-Secure access

These can be configured in the Network Interconnect (NIC) or in the component itself, and the default state is Secure access only from reset.

In general, unless explicitly stated otherwise:

- When a region maps a peripheral or device that occupies less than the region size used, access to the unmapped region results in a DECERR response. For example, when a peripheral occupies 4KB from the 64KB region that is reserved for it.
- Accesses to reserved areas within the memory map also result in a DECERR response. When accessing areas that peripherals or devices occupy, the peripherals or devices themselves determine the response to return. These areas can include unmapped or reserved areas within the areas that the peripheral or device occupies.

The following table summarizes the Lumex address map.

Table 9-1: Lumex Address map

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
RAM/ROM	0x00_0000_0000	0x00_0003_FFFF	AP Secure Boot RAM	256 KB	S	-
RAM/ROM	0x00_0004_0000	0x00_03FF_FFFF	RESERVED for RAM/ROM	63.8 MB	S	-
RAM/ROM	0x00_0400_0000	0x00_0407_FFFF	Secure RAM	512 KB	S	-
RAM/ROM	0x00_0408_0000	0x00_04FF_FFFF	RESERVED for RAM/ROM	15.5 MB	S	-
RAM/ROM	0x00_0500_0000	0x00_0507_FFFF	RESERVED for RAM/ROM	512 KB	S	-
RAM/ROM	0x00_0508_0000	0x00_05FF_FFFF	RESERVED for RAM/ROM	15.5 MB	S	-

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
RAM/ROM	0x00_0600_0000	0x00_0600_FFFF	Non-Secure RAM	64 KB	N/A	-
RAM/ROM	0x00_0601_0000	0x00_07FF_FFFF	RESERVED for RAM/ROM	31.9 MB	N/A	-
RESERVED	0x00_0800_0000	0x00_17FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	256 MB	N/A	-
RESERVED	0x00_1800_0000	0x00_1FFF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	128 MB	N/A	-
RESERVED	0x00_2000_0000	0x00_20FF_FFFF	RESERVED	16 MB	N/A	-
DMC expansion	0x00_2100_0000	0x00_2101_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2102_0000	0x00_2103_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2104_0000	0x00_2107_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	256 KB	N/A	-
DMC expansion	0x00_2108_0000	0x00_2108_0FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-
DMC expansion	0x00_2108_1000	0x00_2108_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-
DMC expansion	0x00_2108_2000	0x00_2109_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	64 KB	N/A	-
DMC expansion	0x00_2109_2000	0x00_210F_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	440 KB	N/A	-
DMC expansion	0x00_2110_0000	0x00_2111_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2112_0000	0x00_2113_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2114_0000	0x00_2117_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	256 KB	N/A	-
DMC expansion	0x00_2118_0000	0x00_2118_0FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-
DMC expansion	0x00_2118_1000	0x00_2118_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-
DMC expansion	0x00_2118_2000	0x00_2119_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	64 KB	N/A	-
DMC expansion	0x00_2119_2000	0x00_211F_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	440 KB	N/A	-
DMC expansion	0x00_2120_0000	0x00_2121_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2122_0000	0x00_2123_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2124_0000	0x00_2127_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	256 KB	N/A	-
DMC expansion	0x00_2128_0000	0x00_2128_0FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
DMC expansion	0x00_2128_1000	0x00_2128_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-
DMC expansion	0x00_2128_2000	0x00_2129_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	64 KB	N/A	-
DMC expansion	0x00_2129_2000	0x00_212F_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	440 KB	N/A	-
DMC expansion	0x00_2130_0000	0x00_2131_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2132_0000	0x00_2133_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	128 KB	N/A	-
DMC expansion	0x00_2134_0000	0x00_2137_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	256 KB	N/A	-
DMC expansion	0x00_2138_0000	0x00_2138_0FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-
DMC expansion	0x00_2138_1000	0x00_2138_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	4 KB	N/A	-
DMC expansion	0x00_2138_2000	0x00_2139_1FFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	64 KB	N/A	-
DMC expansion	0x00_2139_2000	0x00_213F_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	440 KB	N/A	-
DMC expansion	0x00_2140_2000	0x00_21FF_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	12 MB	N/A	-
DMC expansion	0x00_2200_0000	0x00_23FF_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	32 MB	N/A	-
DMC expansion	0x00_2400_0000	0x00_25FF_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	32 MB	N/A	-
DMC expansion	0x00_2600_0000	0x00_27FF_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	32 MB	N/A	-
DMC expansion	0x00_2800_0000	0x00_29FF_FFFF	RESERVED. Used for AXI5_M_DMC<0-3>_CFG_*.	32 MB	N/A	-
RESERVED	0x00_2A00_0000	0x00_2A3F_FFFF	RESERVED	4 MB	N/A	-
Base Peripherals	0x00_2A40_0000	0x00_2A40_FFFF	Non-Secure UART for Base peripherals	64 KB	N/A	-
Base Peripherals	0x00_2A41_0000	0x00_2A41_FFFF	Secure UART for Base peripherals	64 KB	S	-
Base Peripherals	0x00_2A42_0000	0x00_2A42_FFFF	RESERVED for Base peripherals	64 KB	N/A	-
Base Peripherals	0x00_2A43_0000	0x00_2A43_FFFF	RESERVED for Base peripherals	64 KB	N/A	-
Base Peripherals	0x00_2A44_0000	0x00_2A44_FFFF	Generic SoC (AP) Watchdog Control for Base peripherals	64 KB	N/A	-
Base Peripherals	0x00_2A45_0000	0x00_2A45_FFFF	Generic SoC (AP) Watchdog Refresh for Base peripherals	64 KB	N/A	-
Base Peripherals	0x00_2A46_0000	0x00_2A47_FFFF	RESERVED for Base peripherals	128 KB	N/A	-

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
Base Peripherals	0x00_2A48_0000	0x00_2A48_FFFF	Secure SoC (AP) Watchdog Control for Base peripherals	64 KB	S	-
Base Peripherals	0x00_2A49_0000	0x00_2A49_FFFF	Secure SoC (AP) Watchdog Refresh for Base peripherals	64 KB	S	-
Base Peripherals	0x00_2A4A_0000	0x00_2A4A_FFFF	RESERVED for Base peripherals	64 KB	S	-
Base Peripherals	0x00_2A4B_0000	0x00_2A7F_FFFF	RESERVED for Base peripherals	3.3 MB	S	-
Base Peripherals	0x00_2A80_0000	0x00_2A80_FFFF	RESERVED for Base peripherals	64 KB	S	-
Base Peripherals	0x00_2A81_0000	0x00_2A81_FFFF	SoC (AP) System Timer - AP_GTCLK_CNTCTLBase for Base peripherals	64 KB	S	-
Base Peripherals	0x00_2A82_0000	0x00_2A82_FFFF	SoC (AP) System Timer (S) - AP_GTCLK_S_CNTBase for Base peripherals	64 KB	S	-
Base Peripherals	0x00_2A83_0000	0x00_2A83_FFFF	SoC (AP) System Timer (NS) - AP_GTCLK_NS_CNTBase for Base peripherals	64 KB	N/A	-
RESERVED	0x00_2A84_0000	0x00_2CBF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	35.8 MB	N/A	-
RESERVED	0x00_2CC0_0000	0x00_2CEF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	3 MB	N/A	-
RESERVED	0x00_2CF0_0000	0x00_2CFF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	1 MB	N/A	-
GPU	0x00_2D00_0000	0x00_2DFF_FFFF	GPU Block	16 MB	N/A	Not accessible by n__cs_axiap
GPU	0x00_2E00_0000	0x00_2FFF_FFFF	RESERVED for GPU Block	32 MB	N/A	-
GIC	0x00_3000_0000	0x00_37FF_FFFF	GIC	128 MB	N/A	-
GIC	0x00_3800_0000	0x00_3EFF_FFFF	RESERVED for GIC	112 MB	N/A	-
SMMU	0x00_3F00_0000	0x00_43FF_FFFF	SMMU0 (GPU)	80 MB	N/A	-
SMMU	0x00_4400_0000	0x00_44FF_FFFF	RESERVED for SMMU	16 MB	N/A	-
SMS	0x00_4500_0000	0x00_48FF_FFFF	SoC Management Subsystem. See SMS	64 MB	N/A	-
SMS	0x00_4900_0000	0x00_4CFF_FFFF	SoC Management Subsystem. See SMS	64 MB	N/A	-
RESERVED	0x00_4D00_0000	0x00_4D0F_FFFF	RESERVED	1 MB	N/A	-
RESERVED	0x00_4D10_0000	0x00_4EFF_FFFF	RESERVED	31 MB	N/A	-
Interconnect	0x00_4F00_0000	0x00_52FF_FFFF	NoC/MCN-data GPV for Interconnect	64 MB	N/A	Depending on the configuration. Expect 16MB used, rest is RESERVED.

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
Interconnect	0x00_5300_0000	0x00_56FF_FFFF	NoC-per GPV for Interconnect	64 MB	N/A	Depending on the configuration. Expect 16MB used, rest is RESERVED.
CSR	0x00_5700_0000	0x00_57FF_FFFF	Control and Status Registers. See CSR	16 MB	N/A	-
Sensor expansion	0x00_5800_0000	0x00_58FF_FFFF	RESERVED for Sensor expansion	16 MB	N/A	-
ClockControl expansion	0x00_5900_0000	0x00_59FF_FFFF	RESERVED for ClockControl expansion	16 MB	N/A	-
RESERVED	0x00_5A00_0000	0x00_5AFF_FFFF	RESERVED	16 MB	N/A	-
RESERVED	0x00_5B00_0000	0x00_5BFF_FFFF	RESERVED	16 MB	N/A	-
RESERVED	0x00_5C00_0000	0x00_5C00_FFFF	RESERVED	64 KB	N/A	-
RESERVED	0x00_5C01_0000	0x00_5C01_FFFF	RESERVED	64 KB	N/A	-
RESERVED	0x00_5C02_0000	0x00_5C02_FFFF	RESERVED	64 KB	N/A	-
RESERVED	0x00_5C03_0000	0x00_5C03_FFFF	RESERVED	64 KB	N/A	-
RESERVED	0x00_5C04_0000	0x00_5C0F_FFFF	RESERVED	768 KB	N/A	-
RESERVED	0x00_5C10_0000	0x00_5C7F_FFFF	RESERVED	7 MB	N/A	-
RESERVED	0x00_5C80_0000	0x00_5CFF_FFFF	RESERVED	8 MB	N/A	-
Debug	0x00_5D00_0000	0x00_5DFF_FFFF	STM	16 MB	N/A	-
Debug	0x00_5E00_0000	0x00_5E7F_FFFF	RESERVED	8 MB	N/A	-
Debug	0x00_5E80_0000	0x00_5EFF_FFFF	RESERVED	8 MB	N/A	-
DSU	0x00_5F00_0000	0x00_5F9F_FFFF	Cluster0 Utility space for DSU	10 MB	N/A	0x01_0000->0x01_ffff and 0x04_0000->0x04_ffff must be accessible by AP cores (for MPAM and AMU registers), but other ranges cannot be. This restriction must be implemented through an APU configuration. Not accessible by m_cs_axiap.
DSU	0x00_5FA0_0000	0x00_5FFF_FFFF	RESERVED for DSU	6 MB	N/A	-
RESERVED	0x00_6000_0000	0x00_67FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	128 MB	N/A	-
RESERVED	0x00_6800_0000	0x00_6FFF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	128 MB	N/A	-
RESERVED	0x00_7000_0000	0x00_73FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 MB	N/A	-

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
RESERVED	0x00_7400_0000	0x00_77FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 MB	N/A	-
RESERVED	0x00_7800_0000	0x00_7BFF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 MB	N/A	-
RESERVED	0x00_7C00_0000	0x00_7FFF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 MB	N/A	-
DRAM	0x00_8000_0000	0x00_FFFF_FFFF	DRAM	2 GiB	N/A	Striped
RESERVED	0x01_0000_0000	0x03_FFFF_FFFF	RESERVED	12 GiB	N/A	-
Debug Top element	0x04_0000_0000	0x04_0000_FFFF	ROM_APP	64 KB	N/A	-
Debug Top element	0x04_0001_0000	0x04_0001_FFFF	STM	64 KB	N/A	-
Debug Top element	0x04_0002_0000	0x04_0002_FFFF	STM Replicator	64 KB	N/A	-
Debug Top element	0x04_0003_0000	0x04_0003_FFFF	STM ETR	64 KB	N/A	-
Debug Top element	0x04_0004_0000	0x04_0004_FFFF	System CTI	64 KB	N/A	-
Debug Top element	0x04_0005_0000	0x04_0005_FFFF	System ETR	64 KB	N/A	-
Debug Top element	0x04_0006_0000	0x04_0006_FFFF	RESERVED (System0 Funnel)	64 KB	N/A	-
Debug Top element	0x04_0007_0000	0x04_0007_FFFF	RESERVED (System1 Funnel)	64 KB	N/A	-
Debug Top element	0x04_0008_0000	0x04_0008_FFFF	RESERVED	64 KB	N/A	-
Debug Top element	0x04_0009_0000	0x04_0009_FFFF	RESERVED	64 KB	N/A	-
Debug Top element	0x04_000A_0000	0x04_000F_FFFF	RESERVED	384 KB	N/A	-
Debug Top element	0x04_0010_0000	0x04_00FF_FFFF	RESERVED	15 MB	N/A	-
Debug Flex	0x04_0100_0000	0x04_0100_FFFF	Flex0 Funnel	64 KB	N/A	-
Debug Flex	0x04_0101_0000	0x04_0101_FFFF	RESERVED (Flex1 Funnel)	64 KB	N/A	-
Debug Flex	0x04_0102_0000	0x04_01FF_FFFF	RESERVED	15.9 MB	N/A	-
Debug	0x04_0200_0000	0x04_02FD_FFFF	DSU dbg	15.9 MB	N/A	-
Debug	0x04_02FE_0000	0x04_02FE_FFFF	DSU ETF	64 KB	N/A	-
Debug	0x04_02FF_0000	0x04_03FF_FFFF	RESERVED	16.1 MB	N/A	-

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
Debug expansion	0x04_0400_0000	0x04_0400_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0401_0000	0x04_0401_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0402_0000	0x04_0402_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0403_0000	0x04_04FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	15.8 MB	N/A	-
Debug expansion	0x04_0500_0000	0x04_0500_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0501_0000	0x04_0501_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0502_0000	0x04_0502_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0503_0000	0x04_05FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	15.8 MB	N/A	-
Debug expansion	0x04_0600_0000	0x04_0600_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0601_0000	0x04_0601_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0602_0000	0x04_0602_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0603_0000	0x04_06FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	15.8 MB	N/A	-
Debug expansion	0x04_0700_0000	0x04_0700_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0701_0000	0x04_0701_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0702_0000	0x04_0702_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	64 KB	N/A	-
Debug expansion	0x04_0703_0000	0x04_07FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	15.8 MB	N/A	-
Debug	0x04_0800_0000	0x04_0FFF_FFFF	RESERVED	128 MB	N/A	-
Debug	0x04_1000_0000	0x04_17FF_FFFF	RESERVED	128 MB	N/A	-
Debug	0x04_1800_0000	0x04_1800_FFFF	RESERVED	64 KB	N/A	-
Debug	0x04_1801_0000	0x04_1EFF_FFFF	RESERVED	111.9 MB	N/A	-
Debug	0x04_1F00_0000	0x04_1F00_FFFF	ROM_DAP	64 KB	N/A	-
Debug	0x04_1F01_0000	0x04_1F01_FFFF	SDC	64 KB	N/A	-
Debug	0x04_1F02_0000	0x04_1F02_FFFF	AP_RSS	64 KB	N/A	-

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
Debug	0x04_1F03_0000	0x04_1F03_FFFF	AP_SCP	64 KB	N/A	-
Debug	0x04_1F04_0000	0x04_1F04_FFFF	AP_SMS	64 KB	N/A	-
Debug	0x04_1F05_0000	0x04_1F07_FFFF	RESERVED	192 KB	N/A	-
Debug	0x04_1F08_0000	0x04_1F08_FFFF	ROM_EDAP	64 KB	N/A	-
Debug	0x04_1F09_0000	0x04_1F09_FFFF	AP_APP	64 KB	N/A	-
Debug	0x04_1F0A_0000	0x04_1F0A_FFFF	AP_MEM	64 KB	N/A	-
Debug	0x04_1F0B_0000	0x04_1F0F_FFFF	RESERVED	320 KB	N/A	-
Debug	0x04_1F10_0000	0x04_1F10_FFFF	AP_GPU	64 KB	N/A	-
Debug	0x04_1F11_0000	0x04_1F17_FFFF	RESERVED	448 KB	N/A	-
Debug	0x04_1F18_0000	0x04_1F18_FFFF	Expansion	64 KB	N/A	-
Debug	0x04_1F19_0000	0x04_1F1F_FFFF	RESERVED	448 KB	N/A	-
Debug	0x04_1F20_0000	0x04_1F20_FFFF	Expansion	64 KB	N/A	-
Debug	0x04_1F21_0000	0x04_1F7F_FFFF	RESERVED	5.9 MB	N/A	-
Debug	0x04_1F80_0000	0x04_1F8F_FFFF	RESERVED	1 MB	N/A	-
Debug	0x04_1F90_0000	0x04_1F90_FFFF	AP_GPU (CS view)	64 KB	N/A	-
Debug	0x04_1F91_0000	0x04_1FFF_FFFF	RESERVED	6.9 MB	N/A	-
Debug	0x04_2000_0000	0x04_3EFF_FFFF	RESERVED	496 MB	N/A	-
Debug expansion	0x04_3F00_0000	0x04_46FF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	128 MB	N/A	-
Debug expansion	0x04_4700_0000	0x04_4EFF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	128 MB	N/A	-
RESERVED	0x04_4F00_0000	0x05_FFFF_FFFF	RESERVED	6.8 GiB	N/A	-
RESERVED	0x06_0000_0000	0x06_7FFF_FFFF	RESERVED	2 GiB	N/A	-
RESERVED	0x06_8000_0000	0x06_FFFF_FFFF	RESERVED	2 GiB	N/A	-
RESERVED	0x07_0000_0000	0x07_7FFF_FFFF	RESERVED	2 GiB	N/A	-
RESERVED	0x07_8000_0000	0x07_FFFF_FFFF	RESERVED	2 GiB	N/A	-
DRAM	0x08_0000_0000	0x08_7FFF_FFFF	HOLE	2 GiB	N/A	Striped

Region	Start address (Hex)	End Address (Hex)	Description	Size	Access Control	Additional information
DRAM	0x08_8000_0000	0x0F_FFFF_FFFF	DRAM	30 GiB	N/A	Striped
RESERVED	0x10_0000_0000	0x3F_FFFF_FFFF	RESERVED	192 GiB	N/A	-
RESERVED	0x40_0000_0000	0x7F_FFFF_FFFF	RESERVED. Used for AXI5_M_CPNSS_SOUTH_PER_*.	256 GiB	N/A	-
DRAM	0x80_0000_0000	0x87_FFFF_FFFF	HOLE	32 GiB	N/A	Striped
DRAM	0x88_0000_0000	0x100_0000_0000	DRAM	480 GiB	N/A	Striped

9.1.1.1 CSR region memory map

Details the Control and Status Registers (CSR) memory map for the Application Processor (AP) system.

The following table summarizes the CSR region memory map for the AP system.

Table 9-2: CSR region memory map

Start address (Hex)	End address (Hex)	Description	Size	Access control	Additional information
0x0000_0000	0x000F_FFFF	CPU CSR	1 MB	N/A	SCP/RSE accesses only. ATU configuration determines S/NS for SCP/RSE accesses.
0x0010_0000	0x001F_FFFF	GPU CSR	1 MB	N/A	SCP/RSE accesses only. ATU configuration determines S/NS for SCP/RSE accesses.
0x0020_0000	0x002F_FFFF	RESERVED for CSR	1 MB	N/A	-
0x0030_0000	0x003F_FFFF	RESERVED for CSR	1 MB	N/A	-
0x0040_0000	0x004F_FFFF	RESERVED for CSR	1 MB	N/A	-
0x0050_0000	0x005F_FFFF	RESERVED for CSR	1 MB	N/A	-
0x0060_0000	0x006F_FFFF	RESERVED for CSR	1 MB	N/A	-
0x0070_0000	0x007F_FFFF	RESERVED for CSR	1 MB	N/A	-
0x0080_0000	0x008F_FFFF	RESERVED	1 MB	N/A	-
0x0090_0000	0x009F_FFFF	RESERVED	1 MB	N/A	-
0x00A0_0000	0x00AF_FFFF	RESERVED	1 MB	N/A	-
0x00B0_0000	0x00BF_FFFF	RESERVED	1 MB	N/A	-
0x00C0_0000	0x00CF_FFFF	RESERVED for CSR	1 MB	N/A	-
0x00D0_0000	0x00DF_FFFF	RESERVED for CSR	1 MB	N/A	-

Start address (Hex)	End address (Hex)	Description	Size	Access control	Additional information
0x00E0_0000	0x00EF_FFFF	RESERVED for CSR	1 MB	N/A	-
0x00F0_0000	0x00F0_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F0_1000	0x00F0_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F1_0000	0x00F1_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F1_1000	0x00F1_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F2_0000	0x00F2_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F2_1000	0x00F2_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F3_0000	0x00F3_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F3_1000	0x00F3_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F4_0000	0x00F4_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F4_1000	0x00F4_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F5_0000	0x00F5_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F5_1000	0x00F5_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F6_0000	0x00F6_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F6_1000	0x00F6_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F7_0000	0x00F7_0FFF	RESERVED for CSR	4 KB	N/A	-
0x00F7_1000	0x00F7_FFFF	RESERVED for CSR	60 KB	N/A	-
0x00F8_0000	0x00FF_FFFF	RESERVED for CSR	512 KB	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-

Start address (Hex)	End address (Hex)	Description	Size	Access control	Additional information
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-

Start address (Hex)	End address (Hex)	Description	Size	Access control	Additional information
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-
0x00FF_FFFF	0x00FF_FFFF	RESERVED for CSR	0	N/A	-

SMS region memory map

Details the SoC Management Subsystem (SMS) memory map for the Application Processor (AP) system.

The following table summarizes the SMS region memory map for the AP system.

Table 9-3: SMS region memory map

Start address	End address	Region	Description	Size	Access control	Access Permission (by static NoC config)
0x00_4500_0000	0x00_4507_FFFF	SMCF RAM	SMCF RAM	512 KB	N/A	SCP/RSE (R/W) + SMCF DMAs (W) only
0x00_4508_0000	0x00_450F_FFFF	SMCF RAM	RESERVED for SMCF RAM	512 KB	N/A	-
0x00_4510_0000	0x00_452F_FFFF	SMCF RAM	RESERVED for SMCF RAM	2 MB	N/A	-
0x00_4530_0000	0x00_4530_0FFF	SMCF RAM	RESERVED for SMCF RAM	4 KB	N/A	-
0x00_4530_1000	0x00_453F_FFFF	SMCF RAM	RESERVED for SMCF RAM	1020 KB	N/A	-
0x00_4540_0000	0x00_454F_FFFF	RESERVED	RESERVED	1 MB	N/A	-
0x00_4550_0000	0x00_455F_FFFF	RESERVED	RESERVED	1 MB	N/A	-
0x00_4560_0000	0x00_457F_FFFF	RESERVED	RESERVED	2 MB	N/A	-
0x00_4580_0000	0x00_4580_0FFF	RESERVED	RESERVED	4 KB	N/A	-
0x00_4580_1000	0x00_4580_1FFF	RESERVED	RESERVED	4 KB	N/A	-
0x00_4580_2000	0x00_4580_2FFF	RESERVED	RESERVED	4 KB	N/A	-
0x00_4580_3000	0x00_4580_3FFF	RESERVED	RESERVED	4 KB	N/A	-
0x00_4580_4000	0x00_458F_FFFF	RESERVED	RESERVED	1008 KB	N/A	-

Start address	End address	Region	Description	Size	Access control	Access Permission (by static NoC config)
0x00_4590_0000	0x00_45BF_FFFF	RESERVED	RESERVED	3 MB	N/A	-
0x00_45C0_0000	0x00_45C0_FFFF	RESERVED	RESERVED	64 KB	N/A	-
0x00_45C1_0000	0x00_45C1_FFFF	RESERVED	RESERVED	64 KB	N/A	-
0x00_45C2_0000	0x00_45CF_FFFF	RESERVED	RESERVED	896 KB	N/A	-
0x00_45D0_0000	0x00_45DF_FFFF	RESERVED	RESERVED	1 MB	N/A	-
0x00_45E0_0000	0x00_45FF_FFFF	RESERVED	RESERVED	2 MB	N/A	-
0x00_4600_0000	0x00_4600_FFFF	S-MHU	AP_SCP_MHU_0_s	64 KB	S	AP and AXIAP-SYS only
0x00_4601_0000	0x00_4601_FFFF	S-MHU	SCP_AP_MHU_0_r	64 KB	S	AP and AXIAP-SYS only
0x00_4602_0000	0x00_4602_FFFF	S-MHU	AP_SCP_MHU_1_s	64 KB	S	AP and AXIAP-SYS only
0x00_4603_0000	0x00_4603_FFFF	S-MHU	SCP_AP_MHU_1_r	64 KB	S	AP and AXIAP-SYS only
0x00_4604_0000	0x00_4604_FFFF	S-MHU	RESERVED	64 KB	S	AP and AXIAP-SYS only
0x00_4605_0000	0x00_4605_FFFF	S-MHU	RESERVED	64 KB	S	AP and AXIAP-SYS only
0x00_4606_0000	0x00_4606_FFFF	S-MHU	RESERVED	64 KB	S	AP and AXIAP-SYS only
0x00_4607_0000	0x00_460F_FFFF	S-MHU	RESERVED	576 KB	S	AP and AXIAP-SYS only
0x00_4610_0000	0x00_4610_FFFF	S-MHU	RESERVED for S-MHU	64 KB	S	-
0x00_4611_0000	0x00_4611_FFFF	S-MHU	RESERVED for S-MHU	64 KB	S	-
0x00_4612_0000	0x00_4612_FFFF	S-MHU	RESERVED for S-MHU	64 KB	S	-
0x00_4613_0000	0x00_4613_FFFF	S-MHU	RESERVED for S-MHU	64 KB	S	-
0x00_4614_0000	0x00_4614_FFFF	S-MHU	RESERVED for S-MHU	64 KB	S	-
0x00_4615_0000	0x00_4615_FFFF	S-MHU	RESERVED for S-MHU	64 KB	S	-
0x00_4616_0000	0x00_4616_FFFF	S-MHU	RESERVED for S-MHU	64 KB	S	-
0x00_4617_0000	0x00_461F_FFFF	S-MHU	RESERVED for S-MHU	576 KB	S	-
0x00_4620_0000	0x00_4620_FFFF	NS-MHU	AP_SCP_MHU_2_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4621_0000	0x00_4621_FFFF	NS-MHU	SCP_AP_MHU_2_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_4622_0000	0x00_4622_FFFF	NS-MHU	AP_SCP_MHU_3_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4623_0000	0x00_4623_FFFF	NS-MHU	SCP_AP_MHU_3_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_4624_0000	0x00_4624_FFFF	NS-MHU	AP_SCP_MHU_4_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4625_0000	0x00_4625_FFFF	NS-MHU	SCP_AP_MHU_4_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_4626_0000	0x00_4626_FFFF	NS-MHU	AP_SCP_MHU_5_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4627_0000	0x00_4627_FFFF	NS-MHU	SCP_AP_MHU_5_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_4628_0000	0x00_4628_FFFF	NS-MHU	AP_SCP_MHU_6_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4629_0000	0x00_4629_FFFF	NS-MHU	SCP_AP_MHU_6_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_462A_0000	0x00_462F_FFFF	NS-MHU	RESERVED	384 KB	N/A	-
0x00_4630_0000	0x00_464F_FFFF	NS-MHU	RESERVED	2 MB	N/A	-
0x00_4650_0000	0x00_466F_FFFF	NS-MHU	RESERVED	2 MB	N/A	-
0x00_4670_0000	0x00_46FF_FFFF	NS-MHU	RESERVED	9 MB	N/A	-

Start address	End address	Region	Description	Size	Access control	Access Permission (by static NoC config)
0x00_4700_0000	0x00_4700_FFFF	GTCounter	GTCLK CNTControlBase	64 KB	S(RW)	-
0x00_4701_0000	0x00_4701_FFFF	GTCounter	GTCLK CNTReadBase	64 KB	S(R) + NS(R)	-
0x00_4702_0000	0x00_470F_FFFF	GTCounter	RESERVED	896 KB	N/A	-
0x00_4710_0000	0x00_4710_FFFF	GTCounter	RESERVED	64 KB	N/A	-
0x00_4711_0000	0x00_471F_FFFF	GTCounter	RESERVED	960 KB	N/A	-
0x00_4720_0000	0x00_47FF_FFFF	GTCounter	RESERVED	14 MB	N/A	-
0x00_4800_0000	0x00_4800_FFFF	MISC	System ID registers	64 KB	N/A	-
0x00_4801_0000	0x00_480F_FFFF	MISC	RESERVED	960 KB	N/A	-
0x00_4810_0000	0x00_481F_FFFF	MISC	RESERVED	1 MB	N/A	-
0x00_4820_0000	0x00_483F_FFFF	MISC	RESERVED	2 MB	N/A	-
0x00_4840_0000	0x00_487F_FFFF	MISC	RESERVED	4 MB	N/A	-
0x00_4880_0000	0x00_48FF_FFFF	MISC	RESERVED	8 MB	N/A	-
0x00_4900_0000	0x00_4900_FFFF	S-MHU	AP_RSS_MHU_0_s	64 KB	S	AP and AXIAP-SYS only
0x00_4901_0000	0x00_4901_FFFF	S-MHU	RSS_AP_MHU_0_r	64 KB	S	AP and AXIAP-SYS only
0x00_4902_0000	0x00_4902_FFFF	S-MHU	AP_RSS_MHU_1_s	64 KB	S	AP and AXIAP-SYS only
0x00_4903_0000	0x00_4903_FFFF	S-MHU	RSS_AP_MHU_1_r	64 KB	S	AP and AXIAP-SYS only
0x00_4904_0000	0x00_4904_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4905_0000	0x00_4905_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4906_0000	0x00_4906_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4907_0000	0x00_490F_FFFF	S-MHU	RESERVED	576 KB	S	-
0x00_4910_0000	0x00_4910_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4911_0000	0x00_4911_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4912_0000	0x00_4912_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4913_0000	0x00_4913_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4914_0000	0x00_4914_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4915_0000	0x00_4915_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4916_0000	0x00_4916_FFFF	S-MHU	RESERVED	64 KB	S	-
0x00_4917_0000	0x00_491F_FFFF	S-MHU	RESERVED	576 KB	S	-
0x00_4920_0000	0x00_4920_FFFF	NS-MHU	AP_RSS_MHU_2_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4921_0000	0x00_4921_FFFF	NS-MHU	RSS_AP_MHU_2_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_4922_0000	0x00_4922_FFFF	NS-MHU	AP_RSS_MHU_3_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4923_0000	0x00_4923_FFFF	NS-MHU	RSS_AP_MHU_3_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_4924_0000	0x00_4924_FFFF	NS-MHU	AP_RSS_MHU_4_s	64 KB	N/A	AP and AXIAP-SYS only
0x00_4925_0000	0x00_4925_FFFF	NS-MHU	RSS_AP_MHU_4_r	64 KB	N/A	AP and AXIAP-SYS only
0x00_4926_0000	0x00_492F_FFFF	NS-MHU	RESERVED	640 KB	N/A	-

Start address	End address	Region	Description	Size	Access control	Access Permission (by static NoC config)
0x00_4930_0000	0x00_493F_FFFF	NS-MHU	RESERVED	1 MB	N/A	-
0x00_4940_0000	0x00_494F_FFFF	NS-MHU	RESERVED	1 MB	N/A	-
0x00_4950_0000	0x00_49FF_FFFF	NS-MHU	RESERVED	11 MB	N/A	-
0x00_4A00_0000	0x00_4AFF_FFFF	MISC	RESERVED	16 MB	N/A	-
0x00_4B00_0000	0x00_4BFF_FFFF	MISC	RESERVED	16 MB	N/A	-
0x00_4C00_0000	0x00_4CFF_FFFF	MISC	RESERVED	16 MB	N/A	-

9.1.2 SCP Memory map

The System Control Processor (SCP) memory map is divided into Secure and Non-secure regions. The memory alternates between Secure and Non-secure regions on 256 Mbyte regions, with only a few address areas exempted from security mapping because they are related to debug functionality.

Access attempts to unmapped areas that are not targeting a peripheral will receive DECERR or SLVERR responses depending on the responding block.

The following sections list the SCP memory maps:

- SMS NIC-400
- SCP Core
- SCP Customization
- SCP Integration
- SCP System Integration

Security attributes

There are security attributes associated with each area of memory. These security attributes are defined as:

NS_PPC
: Non-secure access only, gated by a PPC.

S_PPC
: Secure access only, gated by a PPC.

S
: Secure access only.

NS
: Non-secure access only.

P
: Privileged access only.

UP
: Unprivileged access allowed.

P_PPC

Unprivileged controlled by PPC.

Exempt

Any access with no security limitations.

S_TGU

Secure access only, gated by a TrustZone Gating Unit (TGU)

NS_TGU

Non-Secure access only, gated by a TrustZone Gating Unit (TGU)

NS_MPC

Secure access only, gated by a Memory Protection Controller (MPC)

S_MPC

Non-Secure access only, gated by a Memory Protection Controller (MPC)

9.1.2.1 SMS NIC-400

The SoC Management Subsystem (SMS) NIC-400 connects the SCP and RSE as well as a number of other peripherals like the SMCF RAM. The address space in this NIC-400 is wider than that in the system address space to enable accesses from the RSE to SCP as well as the connection of the RSE to its private peripherals without exposing them to the system address space. Although the RSE can access system address space through the SCP and the SCP ATU, this is not recommended because the RSE firmware is not designed to configure the SCP ATU.

This section captures how the uppermost bits in the address space are used to control routing between these SMS local targets and the system address space.

The NoC Interface of the SCP and RSE passes through an ATU capable of setting 52 bits of output address. The top 4 bits of this are the Integration Layer Access Identifier and are used by interconnects to determine how the bottom 48 bits should be interpreted. These are decoded as follows:

Table 9-4: SMS NIC-400

Target	SCP ATU	RSE ATU	AP/SYS
SCP 4G M-class address space	-	0x01_0000_0000_0000 to 0x01_0000_FFFF_FFFF	-
RSE A4G M-class address space	-	-	-
AP system memory map	0x00_0000_0000_0000 to 0x00_00FF_FFFF_FFFF	0x00_0000_0000_0000 to 0x00_00FF_FFFF_FFFF	-
AP system memory map (through TBU)	-	0x02_0000_0000_0000 to 0x02_00FF_FFFF_FFFF	-
SMS Components in System Address Space (MHUs, counters, and so on)	0x00_0000_4500_0000 to 0x00_0000_4CFF_FFFF	0x00_0000_4500_0000 to 0x00_0000_4CFF_FFFF	0x00_0000_4500_0000 to 0x00_0000_4CFF_FFFF
RSE private peripherals	-	0x02_0000_0000_0000 to 0x02_00FF_FFFF_FFFF	-

9.1.2.2 SCP core

Describes address space required for the underlying IoT subsystems and SCPs are built.

For more information on the SCP core memory map, see the *Arm® Corstone™ Reference Systems Architecture Specification Ma1*.

9.1.2.3 SCP customization

Describes the address space of the SCP core subsystem. It represents the complete SCP architecture and only excludes extensions required for the integration into the wider system.

The following tables summarize the SCP customized memory maps for Interconnect expansions:

- Table 9-5: Manager AHB Peripheral Interconnect Expansion Interfaces Low Latency on page 90
- Table 9-6: Manager AHB Peripheral Interconnect Expansion Interface High Latency on page 91
- Table 9-7: Manager AHB Peripheral Interconnect Expansion Interfaces Low Latency on page 92
- Table 9-8: Manager AHB Peripheral Interconnect Expansion Interfaces High Latency on page 93

For other memory regions of the SCP core subsystem, see the *Arm® Corstone™ Reference Systems Architecture Specification Ma1*.

Table 9-5: Manager AHB Peripheral Interconnect Expansion Interfaces Low Latency

Start address	End address	Size	Access control	Description	PPC information
0x4010_0000	0x4010_0FFF	4 KB	NS	RESERVED	-
0x4010_1000	0x4010_1FFF	4 KB	NS	RESERVED	-
0x4010_2000	0x4015_2FFF	324 KB	NS	RESERVED	-
0x4015_3000	0x4015_3FFF	4 KB	NS	RESERVED	-
0x4015_4000	0x4015_7FFF	16 KB	NS	RESERVED	-
0x4015_8000	0x4015_8FFF	4 KB	NS	RESERVED	-
0x4015_9000	0x4015_9FFF	4 KB	NS	RESERVED	-
0x4015_A000	0x4015_FFFF	24 KB	NS	RESERVED	-
0x4016_0000	0x4016_FFFF	64 KB	NS_PPC, P_PPC	SCP_AP_MHU_0_s	PERIPHPPCEXP0[1]
0x4017_0000	0x4017_FFFF	64 KB	NS_PPC, P_PPC	AP_SCP_MHU_0_r	PERIPHPPCEXP0[1]
0x4018_0000	0x4018_FFFF	64 KB	NS_PPC, P_PPC	SCP_AP_MHU_1_s	PERIPHPPCEXP0[2]
0x4019_0000	0x4019_FFFF	64 KB	NS_PPC, P_PPC	AP_SCP_MHU_1_r	PERIPHPPCEXP0[2]

Start address	End address	Size	Access control	Description	PPC information
0x401A_0000	0x401A_FFFF	64 KB	NS_PPC, P_PPC	SCP_AP_MHU_2_s	PERIPHPPCEXP0[3]
0x401B_0000	0x401B_FFFF	64 KB	NS_PPC, P_PPC	AP_SCP_MHU_2_r	PERIPHPPCEXP0[3]
0x401C_0000	0x401C_FFFF	64 KB	NS_PPC, P_PPC	SCP_AP_MHU_3_s	PERIPHPPCEXP0[4]
0x401D_0000	0x401D_FFFF	64 KB	NS_PPC, P_PPC	AP_SCP_MHU_3_r	PERIPHPPCEXP0[4]
0x401E_0000	0x401E_FFFF	64 KB	NS_PPC, P_PPC	SCP_AP_MHU_4_s	PERIPHPPCEXP0[5]
0x401F_0000	0x401F_FFFF	64 KB	NS_PPC, P_PPC	AP_SCP_MHU_4_r	PERIPHPPCEXP0[5]
0x4020_0000	0x4020_FFFF	64 KB	NS_PPC, P_PPC	SCP_AP_MHU_5_s	PERIPHPPCEXP0[6]
0x4021_0000	0x4021_FFFF	64 KB	NS_PPC, P_PPC	AP_SCP_MHU_5_r	PERIPHPPCEXP0[6]
0x4022_0000	0x4022_FFFF	64 KB	NS_PPC, P_PPC	SCP_AP_MHU_6_s	PERIPHPPCEXP0[7]
0x4023_0000	0x4023_FFFF	64 KB	NS_PPC, P_PPC	AP_SCP_MHU_6_r	PERIPHPPCEXP0[7]
0x4024_0000	0x4024_FFFF	64 KB	NS	RESERVED	-
0x4025_0000	0x4025_FFFF	64 KB	NS	RESERVED	-
0x4026_0000	0x4026_FFFF	64 KB	NS_PPC, P_PPC	SCP1_SCP0_MHU_0_r	PERIPHPPCEXP0[8]
0x4027_0000	0x4027_FFFF	64 KB	NS_PPC, P_PPC	SCP0_SCP1_MHU_0_s P2P Expansion to Remote SCP	PERIPHPPCEXP0[9]
0x4028_0000	0x4028_FFFF	64 KB	NS	RESERVED	-
0x4029_0000	0x4029_FFFF	64 KB	NS	RESERVED	-
0x402A_0000	0x402A_FFFF	64 KB	NS	RESERVED	-
0x402B_0000	0x402B_FFFF	64 KB	NS	RESERVED	-
0x402C_0000	0x4228_FFFF	31,813 MB	NS	RESERVED	-
0x4229_0000	0x4238_FFFF	1 MB	NS	RESERVED	-
0x4239_0000	0x4248_FFFF	1 MB	NS	RESERVED	-
0x4249_0000	0x4288_FFFF	4 MB	NS	RESERVED	-
0x4289_0000	0x47FF_FFFF	87,438 MB	NS	RESERVED	-

Table 9-6: Manager AHB Peripheral Interconnect Expansion Interface High Latency

Start address	End address	Size	Access control	Description
0x4810_0000	0x4810_FFFF	64 KB	NS	RESERVED
0x4811_0000	0x4811_FFFF	64 KB	NS	RESERVED
0x4812_0000	0x4816_FFFF	320 KB	NS	RESERVED
0x4817_0000	0x4817_0FFF	4 KB	NS	RESERVED

Start address	End address	Size	Access control	Description
0x4817_1000	0x4817_1FFF	4 KB	NS	RESERVED
0x4817_2000	0x49FF_FFFF	30,555 MB	NS	RESERVED
0x4A00_0000	0x4FFF_FFFF	96 MB	NS	Expansion to SCP Integration Layer

Table 9-7: Manager AHB Peripheral Interconnect Expansion Interfaces Low Latency

Start address	End address	Size	Access control	Description	PPC information
0x5010_0000	0x5010_0FFF	4 KB	S	RESERVED	-
0x5010_1000	0x5010_1FFF	4 KB	S	RESERVED	-
0x5010_2000	0x5013_FFFF	248 KB	S	RESERVED	-
0x5014_0000	0x5014_FFFF	64 KB	S	RESERVED	-
0x5015_0000	0x5015_0FFF	4 KB	S_PPC, P_PPC	ATU	PERIPHPPCEXP0[0]
0x5015_1000	0x5015_1FFF	4 KB	S	RESERVED	-
0x5015_2000	0x5015_2FFF	4 KB	S	RESERVED	-
0x5015_3000	0x5015_3FFF	4 KB	S	RESERVED	-
0x5015_4000	0x5015_7FFF	16 KB	S	RESERVED	-
0x5015_8000	0x5015_8FFF	4 KB	S	RESERVED	-
0x5015_9000	0x5015_9FFF	4 KB	S	RESERVED	-
0x5015_A000	0x5015_AFFF	4 KB	S	RESERVED	-
0x5015_B000	0x5015_BFFF	4 KB	S	RESERVED	-
0x5015_C000	0x5015_CFFF	4 KB	S	RESERVED	-
0x5015_D000	0x5015_DFFF	4 KB	S	RESERVED	-
0x5015_E000	0x5015_FFFF	8 KB	S	RESERVED	-
0x5016_0000	0x5016_FFFF	64 KB	S_PPC, P_PPC	SCP_AP_MHU_0_s	PERIPHPPCEXP0[1]
0x5017_0000	0x5017_FFFF	64 KB	S_PPC, P_PPC	AP_SCP_MHU_0_r	PERIPHPPCEXP0[1]
0x5018_0000	0x5018_FFFF	64 KB	S_PPC, P_PPC	SCP_AP_MHU_1_s	PERIPHPPCEXP0[2]
0x5019_0000	0x5019_FFFF	64 KB	S_PPC, P_PPC	AP_SCP_MHU_1_r	PERIPHPPCEXP0[2]
0x501A_0000	0x501A_FFFF	64 KB	S_PPC, P_PPC	SCP_AP_MHU_2_s	PERIPHPPCEXP0[3]
0x501B_0000	0x501B_FFFF	64 KB	S_PPC, P_PPC	AP_SCP_MHU_2_r	PERIPHPPCEXP0[3]
0x501C_0000	0x501C_FFFF	64 KB	S_PPC, P_PPC	SCP_AP_MHU_3_s	PERIPHPPCEXP0[4]
0x501D_0000	0x501D_FFFF	64 KB	S_PPC, P_PPC	AP_SCP_MHU_3_r	PERIPHPPCEXP0[4]
0x501E_0000	0x501E_FFFF	64 KB	S_PPC, P_PPC	SCP_AP_MHU_4_s	PERIPHPPCEXP0[5]
0x501F_0000	0x501F_FFFF	64 KB	S_PPC, P_PPC	AP_SCP_MHU_4_r	PERIPHPPCEXP0[5]

Start address	End address	Size	Access control	Description	PPC information
0x5020_0000	0x5020_FFFF	64 KB	S_PPC, P_PPC	SCP_AP_MHU_5_s	PERIPHPPCEXP0[6]
0x5021_0000	0x5021_FFFF	64 KB	S_PPC, P_PPC	AP_SCP_MHU_5_r	PERIPHPPCEXP0[6]
0x5022_0000	0x5022_FFFF	64 KB	S_PPC, P_PPC	SCP_AP_MHU_6_s	PERIPHPPCEXP0[7]
0x5023_0000	0x5023_FFFF	64 KB	S_PPC, P_PPC	AP_SCP_MHU_6_r	PERIPHPPCEXP0[7]
0x5024_0000	0x5024_FFFF	64 KB	S	RESERVED	-
0x5025_0000	0x5025_FFFF	64 KB	S	RESERVED	-
0x5026_0000	0x5026_FFFF	64 KB	S_PPC, P_PPC	SCP1_SCP0_MHU_0_r	PERIPHPPCEXP0[8]
0x5027_0000	0x5027_FFFF	64 KB	S_PPC, P_PPC	SCP0_SCP1_MHU_0_s P2P Expansion to Remote SCP	PERIPHPPCEXP0[9]
0x5028_0000	0x57FF_FFFF	125,5 MB	S	RESERVED	-

Table 9-8: Manager AHB Peripheral Interconnect Expansion Interfaces High Latency

Start address	End address	Size	Access control	Description	PPC information
0x5810_0000	0x5810_FFFF	64 KB	S_PPC, P_PPC	SLOWCLK CNTControl	PERIPHPPCEXP0[10]
0x5811_0000	0x5811_FFFF	64 KB	S	RESERVED	-
0x5812_0000	0x59FF_FFFF	30,875 MB	S	RESERVED	-
0x5A00_0000	0x5FFF_FFFF	96 MB	S	Expansion to SCP Integration Layer	-

9.1.2.4 SCP integration

Describes the address space of the SCP integration layer. It enables peripherals that are generally required for the integration of SCP into any SoC platform.

The following tables summarize the SCP Integration Layer memory maps.

Table 9-9: SCP AXI Manager Expansion Interface High Latency

Start address	End address	Size	Access Control	Description
0xA000_0000	0xA7FF_FFFF	128 MB	NS	RESERVED
0xA800_0000	0xAFFF_FFFF	128 MB	NS	Mapped for SCP Expansion
0xB000_0000	0xB7FF_FFFF	128 MB	S	RESERVED
0xB800_0000	0xBFFF_FFFF	128 MB	S	Mapped for SCP Expansion

Table 9-10: Manager AHB Peripheral Interconnect Expansion Interface High Latency (HMSTEXPPIHL)

Start address	End address	Size	Access Control	Description	PPC Information
0x4A00_0000	0x4A00_FFFF	64 KB	NS_PPC, P_PPC	SCP_RSS_MHU_0_s	PERIPHPPCEXP2[0]
0x4A01_0000	0x4A01_FFFF	64 KB	-	RSS_SCP_MHU_0_r	PERIPHPPCEXP2[0]

Start address	End address	Size	Access Control	Description	PPC Information
0x4A02_0000	0x4A02_FFFF	64 KB	NS_PPC, P_PPC	RESERVED	-
0x4A03_0000	0x4A03_FFFF	64 KB	NS	RESERVED	-
0x4A04_0000	0x4A04_0FFF	4 KB	NS_PPC, P_PPC	UART	PERIPHPPCEXP2[1]
0x4A04_1000	0x4A04_1FFF	4 KB	NS_PPC, P_PPC	GPIO (NS)	PERIPHPPCEXP2[2]
0x4A04_2000	0x4A04_2FFF	4 KB	NS	RESERVED	-
0x4A04_3000	0x4A04_3FFF	4 KB	NS	RESERVED	-
0x4A04_4000	0x4A7F_FFFF	7,734 MB	NS	RESERVED	-
0x4A80_0000	0x4A8F_FFFF	1 MB	NS	AON CSR	PERIPHPPCEXP2[4]
0x4A90_0000	0x4AFF_FFFF	7 MB	NS	RESERVED	-
0x4B00_0000	0x4BFF_FFFF	16 MB	NS	SCP System integration	-
0x4C00_0000	0x4FFF_FFFF	64 MB	NS	SCP Expansion	-
0x5A00_0000	0x5A00_FFFF	64 KB	S_PPC, P_PPC	SCP_RSS_MHU_0_s	PERIPHPPCEXP2[0]
0x5A01_0000	0x5A01_FFFF	64 KB	-	RSS_SCP_MHU_0_r	PERIPHPPCEXP2[0]
0x5A02_0000	0x5A02_FFFF	64 KB	S_PPC, P_PPC	RESERVED	-
0x5A03_0000	0x5A03_FFFF	64 KB	S	RESERVED	-
0x5A04_0000	0x5A04_0FFF	4 KB	S_PPC, P_PPC	UART	PERIPHPPCEXP2[1]
0x5A04_1000	0x5A04_1FFF	4 KB	S	RESERVED	-
0x5A04_2000	0x5A04_2FFF	4 KB	S_PPC, P_PPC	GPIO (S)	PERIPHPPCEXP2[3]
0x5A04_3000	0x5A04_3FFF	4 KB	S_PPC, P_PPC	RESERVED	-
0x5A04_4000	0x5A7F_FFFF	7,734 MB	S	RESERVED	-
0x5A80_0000	0x5A8F_FFFF	1 MB	S	AON CSR	PERIPHPPCEXP2[4]
0x5A90_0000	0x5AFF_FFFF	7 MB	S	RESERVED	-
0x5B00_0000	0x5BFF_FFFF	16 MB	S	SCP System integration	-
0x5C00_0000	0x5FFF_FFFF	64 MB	S	SCP Expansion	-

9.1.2.5 SCP system integration

Describes the address space of the SCP System Integration Layer. It enables peripherals that are specific for the integration specific to the TCSS-24 platform.

The following tables summarize the SCP System Integration Layer memory maps.

Table 9-11: Manager AHB Peripheral Interconnect Expansion Interface High Latency (HMSTEXPPIHL)

Start address	End address	Size	Access Control	Description	PPC Information
0x4B00_0000	0x4B0F_FFFF	1 MB	NS_PPC, P_PPC	RESERVED	-
0x4B10_0000	0x4B1F_FFFF	1 MB	NS_PPC, P_PPC	SYS CSR	PERIPHPPCEXP2[5]
0x4B20_0000	0x4B7F_FFFF	6 MB	NS	RESERVED	-
0x4B80_0000	0x4B80_FFFF	64 KB	NS_PPC, P_PPC	SYSTOP0_PPU	PERIPHPPCEXP2[6]
0x4B81_0000	0x4B81_FFFF	64 KB	NS_PPC, P_PPC	SYSTOP1_PPU	PERIPHPPCEXP2[7]
0x4B82_0000	0x4B8F_FFFF	896 KB	NS	RESERVED	-
0x4B90_0000	0x4B90_FFFF	64 KB	NS_PPC, P_PPC	RESERVED	-

Start address	End address	Size	Access Control	Description	PPC Information
0x4B91_0000	0x4B91_FFFF	64 KB	NS_PPC, P_PPC	RESERVED	-
0x4B92_0000	0x4BFF_FFFF	6,875 MB	NS	RESERVED	-
0x5B00_0000	0x5B0F_FFFF	1 MB	S_PPC, P_PPC	RESERVED	-
0x5B10_0000	0x5B1F_FFFF	1 MB	S_PPC, P_PPC	SYS CSR	PERIPHPPCEXP2[5]
0x5B20_0000	0x5B7F_FFFF	6 MB	S	RESERVED	-
0x5B80_0000	0x5B80_FFFF	64 KB	S_PPC, P_PPC	SYSTOP0_PPU	PERIPHPPCEXP2[6]
0x5B81_0000	0x5B81_FFFF	64 KB	S_PPC, P_PPC	SYSTOP1_PPU	PERIPHPPCEXP2[7]
0x5B82_0000	0x5B8F_FFFF	896 KB	S	RESERVED	-
0x5B90_0000	0x5B90_FFFF	64 KB	S_PPC, P_PPC	RESERVED	-
0x5B91_0000	0x5B91_FFFF	64 KB	S_PPC, P_PPC	RESERVED	-
0x5B92_0000	0x5BFF_FFFF	6,875 MB	S	RESERVED	-

9.1.3 RSE Memory map

The RSE memory map is divided into Secure and Non-Secure regions. The memory alternates between Secure and Non Secure regions on 256Mbyte regions, with only a few address areas exempted from security mapping because they are related to debug functionality.

Access attempts to unmapped areas that are not targeting a peripheral will receive DECERR or SLVERR responses depending on the responding block.

The following sections describe the actual Lumex implementation for RSE Expansion regions:

- RSE Customization
- RSE Integration

9.1.3.1 RSE Customization

Describes the address space of the RSE core subsystem. It represents the complete RSE architecture and only excludes extensions required for the integration into the wider system.

9.1.3.2 RSE Integration

Describes the address space of the RSE Integration Layer. It covers peripherals that are generally required for the integration of RSE into any SoC platform.

The following tables summarize the RSE Integration Layer memory maps.

Table 9-12: RSE AXI Interconnect Interface (MCI in RTL)

Start address	End address	Access Control	Size	Description
0x0200_0000	0x02FF_FFFF	NS	16 MB	Allocated for non-Secure code execution through SIC and ATU
0x6000_0000	0x67FF_FFFF	NS	128 MB	Allocated for ATU non-Secure data access

Start address	End address	Access Control	Size	Description
0x6800_0000	0x6FFF_FFFF	NS	128 MB	RESERVED (bus error)
0x1200_0000	0x12FF_FFFF	S	16 MB	Allocated for Secure code execution through SIC and ATU
0x7000_0000	0x77FF_FFFF	S	128 MB	Allocated for ATU Secure data access
0x7800_0000	0x7FFF_FFFF	S	128 MB	RESERVED (bus error)

Table 9-13: RSE AXI Manager Integration Expansion Interface (EXMI in RTL)

Start address	End address	Access Control	Size	Description
0x2800_0000	0x2BFF_FFFF	NS	64 MB	RESERVED (bus error)
0x2C00_0000	0x2FFF_FFFF	NS	64 MB	Allocated for non-Secure RSE Manager AXI Integration Expansion Interface.
0x8000_0000	0x87FF_FFFF	NS	128 MB	RESERVED (bus error)
0x8800_0000	0x8FFF_FFFF	NS	128 MB	Allocated for non-Secure RSE Manager AXI Integration Expansion Interface.
0xA000_0000	0xA7FF_FFFF	NS	128 MB	RESERVED (bus error)
0xA800_0000	0xAFFF_FFFF	NS	128 MB	Allocated for non-Secure RSE Manager AXI Integration Expansion Interface.
0xC000_0000	0xC7FF_FFFF	NS	128 MB	RESERVED (bus error)
0xC800_0000	0xCFFF_FFFF	NS	128 MB	Allocated for non-Secure RSE Manager AXI Integration Expansion Interface.
0x3800_0000	0x3BFF_FFFF	S	64 MB	RESERVED (bus error)
0x3C00_0000	0x3FFF_FFFF	S	64 MB	Allocated for Secure RSE Manager AXI Integration Expansion Interface.
0x9000_0000	0x97FF_FFFF	S	128 MB	RESERVED (bus error)
0x9800_0000	0x9FFF_FFFF	S	128 MB	Allocated for Secure RSE Manager AXI Integration Expansion Interface.
0xB000_0000	0xB7FF_FFFF	S	128 MB	RESERVED (bus error)
0xB800_0000	0xBFFF_FFFF	S	128 MB	Allocated for Secure RSE Manager AXI Integration Expansion Interface.
0xD000_0000	0xD7FF_FFFF	S	128 MB	RESERVED (bus error)
0xD800_0000	0xDFFF_FFFF	S	128 MB	Allocated for Secure RSE Manager AXI Integration Expansion Interface.

Table 9-14: RSE AHB Manager Peripheral Integration Expansion Interface (EXMPI in RTL)

Start address	End address	Access Control	Size	Description	PPC Information
0x4810_0000	0x4810_0FFF	NS	4 KB	UART	PERIPHPPCEXP2[0]
0x4810_1000	0x481F_FFFF	NS	0,996 MB	RESERVED (bus error)	-
0x4820_0000	0x4820_FFFF	NS	64 KB	Local X-Die MHU	PERIPHPPCEXP2[1]
0x4821_0000	0x4821_FFFF	NS	64 KB	Distant X-Die MHU	PERIPHPPCEXP2[2]
0x4822_0000	0x487F_FFFF	NS	5,875 MB	RESERVED (bus error)	-
0x4880_0000	0x488F_FFFF	NS	1 MB	RSE IL CSR	PERIPHPPCEXP2[3]
0x4890_0000	0x489F_FFFF	NS	1 MB	RESERVED (bus error)	-
0x48A0_0000	0x48AF_FFFF	NS	1 MB	RESERVED (bus error)	-
0x48B0_0000	0x48FF_FFFF	NS	5 MB	RESERVED (bus error)	-
0x4900_0000	0x4BFF_FFFF	NS	48 MB	RESERVED (bus error)	-
0x4C00_0000	0x4FFF_FFFF	NS	64 MB	Mapped for RSE expansion	-
0xE020_0000	0xE7FF_FFFF	NS	126 MB	RESERVED (bus error)	-
0xE800_0000	0xEFFF_FFFF	NS	128 MB	Mapped for RSE expansion	-
0x5810_0000	0x5810_0FFF	S	4 KB	UART	PERIPHPPCEXP2[0]
0x5810_1000	0x581F_FFFF	S	0,996 MB	RESERVED (bus error)	-

Start address	End address	Access Control	Size	Description	PPC Information
0x5820_0000	0x5820_FFFF	S	64 KB	Local X-Die MHU	PERIPHPPCEXP2[1]
0x5821_0000	0x5821_FFFF	S	64 KB	Distant X-Die MHU	PERIPHPPCEXP2[2]
0x5822_0000	0x587F_FFFF	S	5,875 MB	RESERVED (bus error)	-
0x5880_0000	0x588F_FFFF	S	1 MB	RSE IL CSR	PERIPHPPCEXP2[3]
0x5890_0000	0x589F_FFFF	S	1 MB	RESERVED (bus error)	-
0x58A0_0000	0x58AF_FFFF	S	1 MB	RESERVED (bus error)	-
0x58B0_0000	0x58FF_FFFF	S	5 MB	RESERVED (bus error)	-
0x5900_0000	0x5BFF_FFFF	S	48 MB	RESERVED (bus error)	-
0x5C00_0000	0x5FFF_FFFF	S	64 MB	Mapped for RSE expansion	-
0xF020_0000	0xF7FF_FFFF	S	126 MB	RESERVED (bus error)	-
0xF800_0000	0xFFFF_FFFF	S	128 MB	Mapped for RSE expansion	-

9.1.4 Debug Memory map

The debug memory map covers the visible address regions at the debug access ports (AP).

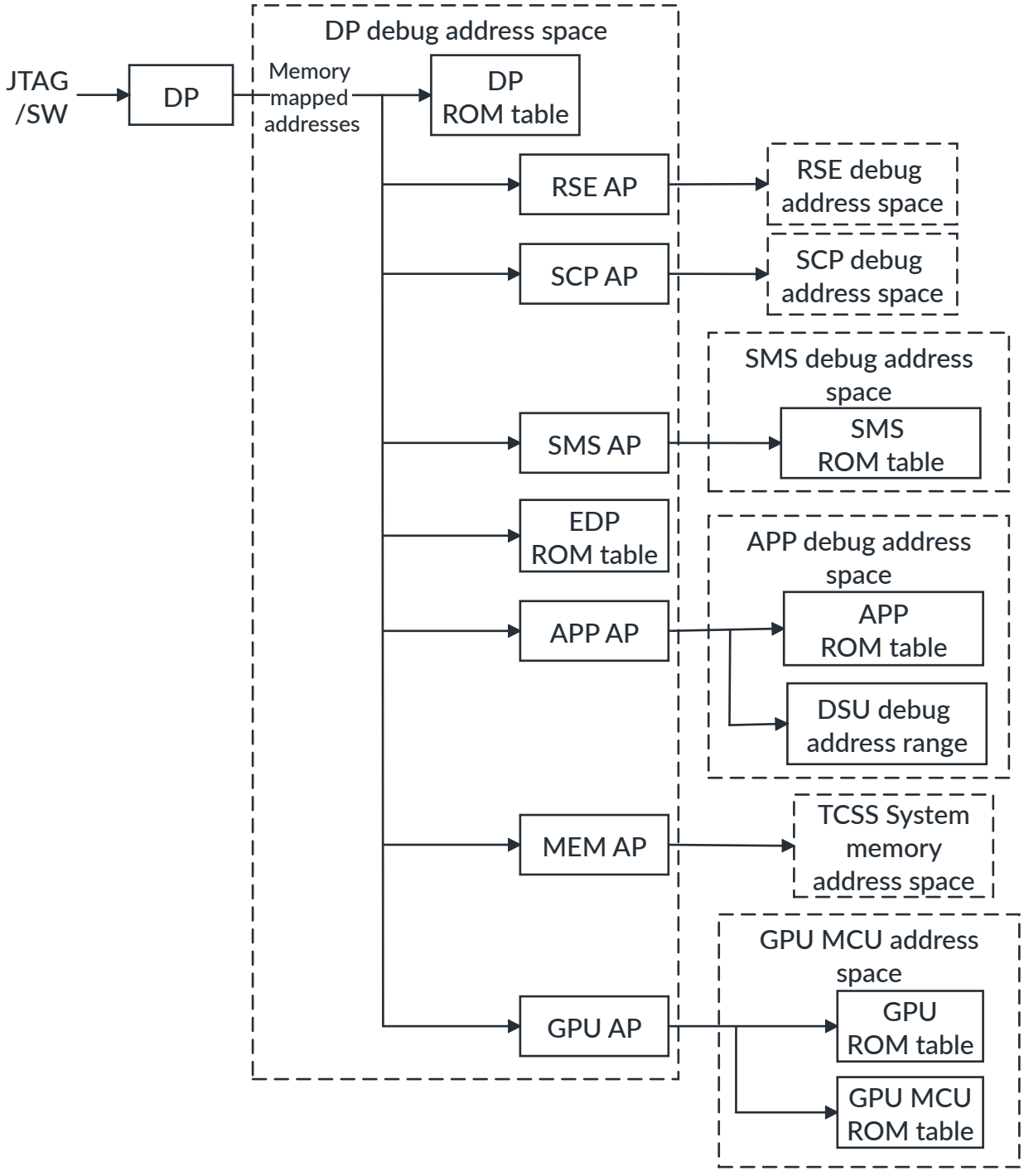
For more information on the mapping of address regions to targets and detail access restrictions where applicable, see the following tables:

- Table 9-15: DP debug address space on page 99
- Table 9-16: SMS debug address space on page 99
- Table 9-17: APP debug address space on page 100
- Table 9-18: GPU debug address space on page 101

An external debug agent is normally a hardware debugger attached to the physical *Joint Test Action Group* (JTAG) debug interface or *Serial Wire* (SW) debug interface. These interfaces are connected to a CoreSight™ Debug Port (DP) which converts JTAG and SW protocol to APB protocol. APB is the main protocol used by debug interconnect.

Figure 9-1: Debug memory map organization viewed from external debug agent on page 98 shows how the debug memory map viewed from the external debug agent is organized in a layered structure.

Figure 9-1: Debug memory map organization viewed from external debug agent



Note

The above diagram only shows the components that affect the organization of the debug memory map. Normal debug components, such as CTI and ETR, are not included inside.

The DP debug address space is directly viewed from the DP, and is the top-layer of the memory map. The following table shows the debug components inside the address space.

Table 9-15: DP debug address space

Debug component	Address offset	Comment
DP ROM + GPR	0x0000_0000	Using css600_apbrom_gpr. The ROM table contains the entries up to EDP ROM + GPR. Apart from the debug ROM table, Its *Granular Power Requestor* (GPR) can request power up of PD_RSE and PD_SCP power domains.
SDC	0x0001_0000	SDC-600
RSE AP	0x0002_0000	AHB-AP
SCP AP	0x0003_0000	AHB-AP
SMS AP	0x0004_0000	AHB-AP
EDP ROM + GPR	0x0008_0000	The ROM table contains the entries below EDP ROM + GPR. Its GPR can request power up of PD_SYSTOP power domain.
APP AP	0x0009_0000	AHB-AP
MEM AP	0x000A_0000	AXI-AP
GPU AP	0x0010_0000	AHB-AP

Note

Each entry in the preceding table takes 64KB address space. Other address spaces are reserved.

An *Access Port* (AP) provides a bridge into a next layer of address space. The following address spaces explained are bridged from APs as listed in Table 9-15: DP debug address space on page 99.

RSE AP

This AP bridges to RSE debug address space which contains the memory-mapped debug components within the RSE. For more information about RSE debug, see the *RSERM?*.

SCP AP

This AP bridges to SCP debug address space which contains the memory-mapped debug components within the SCP. For more information about SCP debug, see the *Arm® Corstone™ Reference Systems Architecture Specification Ma1*.

SMS AP

This AP bridges to SMS debug address space which contains the memory-mapped debug components within the Debug AON part of the Debug Block. The following table shows debug components inside this address space.

Table 9-16: SMS debug address space

Debug component	Address offset	Comment
SMS ROM	0x0000_0000	Using css600_apbrom. The ROM table contains the entries of memory-mapped debug components in SMS debug address space.

Debug component	Address offset	Comment
SMS ATB funnel	0x0001_0000	-
SMS ATB replicator	0x0002_0000	-
SMS CTI	0x0003_0000	-
SMS TPIU	0x0004_0000	-

APP AP

This AP bridges to APP debug address space which contains the memory-mapped debug components in the Processor Block and the memory-mapped debug components within the Debug SYS part of the Debug Block. The following table shows the debug components inside this address space.

Table 9-17: APP debug address space

Debug component	Address offset	Comment
APP ROM + GPR	0x0000_0000	Using css600_apbrom_gpr. The ROM table contains the entries of memory-mapped debug components in APP debug address space. Apart from the debug ROM table, Its GPR can request power up of processor related power domains.
STM	0x0001_0000	-
STM ATB replicator	0x0002_0000	-
STM ETR	0x0003_0000	-
SYS CTI	0x0004_0000	-
SYS ETR	0x0005_0000	-
SYS ATB funnel	0x0006_0000	-
Flex ATB funnel	0x0100_0000	-
DSU debug address range	0x0200_0000	The address range is mapped to DUS debug ports. This offset points to the first ROM table inside DSU. For more information about DSU debug, see *Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual*.
DSU ETF	0x02FE_0000	This *Embedded Trace FIFO* (ETF) is added in Processor Block for smoothing out a bursty trace from DSU.
Reserved debug address range for DMC0	0x0400_0000	The address range is mapped to the debug expansion for DMC0.
Reserved debug address range for DMC1	0x0500_0000	The address range is mapped to the debug expansion for DMC1.
Reserved debug address range for DMC2	0x0600_0000	The address range is mapped to the debug expansion for DMC2.
Reserved debug address range for DMC3	0x0700_0000	The address range is mapped to the debug expansion for DMC3.

MEM AP

This AP bridges to the System memory map of Lumex. It gives an external debug agent the access of the memory space.

GPU AP

This AP bridges to Mali G1-Ultra MCU address space which contains the memory-mapped debug components within the GPU. The following table shows the debug components inside this address space.

Table 9-18: GPU debug address space

Debug component	Address offset	Comment
GPU ROM	`0xE005_0000`	Using css600_apbrom. The ROM table contains the entries of memory-mapped debug components in GPU debug address space.
GPU ETF	`0xE005_1000`	-
Mali G1-Ultra MCU ROM	`0xE00F_E000`	This address offset points to the GPU MCU ROM table, from which the GPU MCU debug components can be located.

ROM tables are used for facilitating auto-discovery of memory-mapped debug components by an external debug agent. In a debug address space, if there are more than one memory-mapped debug components, a ROM table is placed at the address specified by the DP or AP of that space. It contains entries which point to other debug components in the space. The debug components can be other ROM tables, such as the EDP ROM table shown in Figure 9-1: Debug memory map organization viewed from external debug agent on page 98. Together with the DP ROM table, they point to all the debug components within the DP Debug address space. A ROM table can optionally contain *Granular Power Requestor* (GPR) functionality, which allows it to request power up of certain power domains. For more information about ROM tables, see *Arm® Debug Interface Architecture Specification ADIv6.0*.

9.2 Interrupt maps

The *Application Processor* (AP), SCP and RSE have their own interrupt maps which are different from each other.

9.2.1 Application processor interrupt map

The Generic Interrupt Controller (GIC) Architecture defines two different types of interrupts. They are Private Peripheral Interrupts (PPIs) and Shared Peripheral Interrupts (SPIs). PPIs separately exist for every processor, while SPIs are shared for all processors.

PPI and SPI interrupts have configurable options, including number of interrupts, Edge or Level triggered, and Polarity. For example, active-LOW, active-HIGH, or Rising edge.

The following table summarizes the interrupt map of the Application Processor (AP).

Note

- GIC PPI inputs are either active-LOW level sensitive, or triggered on a rising edge.
- Interrupt ID range from 0-31 is for SGI and PPI interrupts. SPI-C interrupt ID range in Lumex starts from 32 as shown in the below table.

Table 9-19: AP interrupt map

Component	Interrupt id	Interrupt source	Trigger	Polarity	Description
DSU	0-15	SGI	-	-	Software Generated Interrupt
DSU	16	RESERVED	-	-	-
DSU	17	pmbirqn	Level	active-LOW	Statistical Profiling Extension interrupt (only for CXC and “big” cores)
DSU	18	trbirqn	Level	active-LOW	Trace Profiling Buffer event
DSU	19	cnthvsirqn	Level	active-LOW	Secure Virtual Timer event
DSU	20	cnthpsirqn	Level	active-LOW	Secure Physical Timer event
DSU	21	RESERVED	-	-	-
DSU	22	commirqn	Level	active-LOW	Debug Communications Channel receive or transmit request
DSU	23	pmuirqn	Level	active-LOW	PMU interrupt
DSU	24	ctiirqn	Edge	Rising edge	CTI interrupt
DSU	25	vcpumntirqn	Level	active-LOW	Virtual Maintenance Interrupt
DSU	26	cnthpirqn	Level	active-LOW	Non-secure PL2 Timer event
DSU	27	cntvirqn	Level	active-LOW	Virtual Timer event
DSU	28	cnthvirqn	Level	active-LOW	-
DSU	29	cntpsirqn	Level	active-LOW	Secure PL1 Physical Timer event
DSU	30	cntpnsirqn	Level	active-LOW	Non-secure PL1 Physical Timer event
DSU	31	RESERVED	-	-	-
SoC Expansion	32-47	RESERVED	-	-	-
MCN	48	MCN0 - Combined S	Level	active-HIGH	irpt_comb_s
MCN	49	MCN0 - Combined NS	Level	active-HIGH	irpt_comb_ns
MCN	50	RESERVED	-	-	-
MCN	51	RESERVED	-	-	-

Component	Interrupt id	Interrupt source	Trigger	Polarity	Description
MCN	52	MCN1 - Combined S	Level	active-HIGH	-
MCN	53	MCN1 - Combined NS	Level	active-HIGH	-
MCN	54	RESERVED	-	-	-
MCN	55	RESERVED	-	-	-
MCN	56	MCN2 - Combined S	Level	active-HIGH	-
MCN	57	MCN2 - Combined NS	Level	active-HIGH	-
MCN	58	RESERVED	-	-	-
MCN	59	RESERVED	-	-	-
MCN	60	MCN3 - Combined S	Level	active-HIGH	-
MCN	61	MCN3 - Combined NS	Level	active-HIGH	-
MCN	62	RESERVED	-	-	-
MCN	63	RESERVED	-	-	-
MCN	64	MCN4 - Combined S	Level	active-HIGH	-
MCN	65	MCN4 - Combined NS	Level	active-HIGH	-
MCN	66	RESERVED	-	-	-
MCN	67	RESERVED	-	-	-
MCN	68	MCN5 - Combined S	Level	active-HIGH	-
MCN	69	MCN5 - Combined NS	Level	active-HIGH	-
MCN	70	RESERVED	-	-	-
MCN	71	RESERVED	-	-	-
MCN	72	MCN6 - Combined S	Level	active-HIGH	-
MCN	73	MCN6 - Combined NS	Level	active-HIGH	-
MCN	74	RESERVED	-	-	-
MCN	75	RESERVED	-	-	-
MCN	76	MCN7 - Combined S	Level	active-HIGH	-
MCN	77	MCN7 - Combined NS	Level	active-HIGH	-
MCN	78	RESERVED	-	-	-
MCN	79	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW
CS	80	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW

Component	Interrupt id	Interrupt source	Trigger	Polarity	Description
CS	81	SystemETR - ETRBUFINT	Level	active-HIGH	From SystemETR
CS	82	STMETR - ETRBUFINT	Level	active-HIGH	From STMETR
CS	83	STM-500 synchronization	Edge	Rising edge	STM-500 Synchronization Interrupt
CS	84	System CTI trigger0 output	Edge	Rising edge	Driven from debug subsystem CTM
CS	85	System CTI trigger1 output	Edge	Rising edge	Driven from debug subsystem CTM
Base Block	86	Secure Watchdog (WS0)	Level	active-HIGH	Bit 0: First Secure WD Expiry
Base Block	87	Secure Watchdog (WS1)	Level	active-HIGH	Bit 1: Second Secure WD Expiry
Base Block	90:88	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW
Base Block	91	AP_GTCLK Generic timer (secure)	Level	active-HIGH	-
Base Block	92	AP_GTCLK Generic timer (non-secure)	Level	active-HIGH	-
Base Block	93	Non-Secure Watchdog WS0	Level	active-HIGH	Bit 0: First Non-secure WD Expiry
Base Block	94	Non-Secure Watchdog WS1	Level	active-HIGH	Bit 1: Second Non-secure WD Expiry
Base Block	95	AP_NS_UART_INT	Level	active-HIGH	-
Base Block	96	AP_S_UART_INT	Level	active-HIGH	-
GIC	97	GIC PMU IRQ	-	-	GIC PMU counter overflow interrupt
GIC	98	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW
GIC	99	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW
GIC	100	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW
DMC Expansion	101-116	RESERVED	-	-	-
DMC Expansion	117-127	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW
COM Expansion	128-255	Expansion interrupts	-	-	SoC Expansion active-HIGH level interrupts. Otherwise reserved and tied LOW
SMMU	256	TCU PMU IRPT	Edge	Rising edge	-
SMMU	257	TCU Event Queue Secure	Edge	Rising edge	-
SMMU	258	TCU CMD SYNC Secure	Edge	Rising edge	-
SMMU	259	TCU Global Secure	Edge	Rising edge	-

Component	Interrupt id	Interrupt source	Trigger	Polarity	Description
SMMU	260	TCU Event Queue Non-secure	Edge	Rising edge	-
SMMU	261	TCU CMD SYNC Non-secure	Edge	Rising edge	-
SMMU	262	TCU Global Non-secure	Edge	Rising edge	-
SMMU	263	RESERVED	Edge	Rising edge	-
SMMU	264	GPU0 TBU PMU interrupt	Edge	Rising edge	-
SMMU	265	GPU1 TBU PMU interrupt	Edge	Rising edge	-
SMMU	266	GPU2 TBU PMU interrupt	Edge	Rising edge	-
SMMU	267	GPU3 TBU PMU interrupt	Edge	Rising edge	-
SMMU	268	SMS SCP/RSE TBU PMU	Edge	Rising edge	-
SMMU	269	SMS SYS ETR TBU PMU	Edge	Rising edge	-
SMMU	270	SMS ETR TBU PMU interrupt	Edge	Rising edge	-
SMMU	271	RESERVED	-	-	-
GPU Block	272	GPU IRQSYS	Level	active-HIGH	-
GPU Block	273	GPU IRQGROUP	Level	active-HIGH	-
GPU Block	274	GPU IRQAW0	Level	active-HIGH	-
GPU Block	275	GPU IRQAW1	Level	active-HIGH	-
GPU Block	276	GPU IRQAW2	Level	active-HIGH	-
GPU Block	277	GPU IRQAW3	Level	active-HIGH	-
GPU Block	278	GPU IRQAW4	Level	active-HIGH	-
GPU Block	279	GPU IRQAW5	Level	active-HIGH	-
GPU Block	280	GPU IRQAW6	Level	active-HIGH	-
GPU Block	281	GPU IRQAW7	Level	active-HIGH	-
GPU Block	282	GPU IRQAW8	Level	active-HIGH	-

Component	Interrupt id	Interrupt source	Trigger	Polarity	Description
GPU Block	283	GPU IRQAW9	Level	active-HIGH	-
GPU Block	284	GPU IRQAW10	Level	active-HIGH	-
GPU Block	285	GPU IRQAW11	Level	active-HIGH	-
GPU Block	286	GPU IRQAW12	Level	active-HIGH	-
GPU Block	287	GPU IRQAW13	Level	active-HIGH	-
GPU Block	288	GPU IRQAW14	Level	active-HIGH	-
GPU Block	289	GPU IRQAW15	Level	active-HIGH	-
DSU	290	pmuirqn	Level	active-HIGH	DSU PMU interrupt
RESERVED	291-319	RESERVED	-	-	Treated as active-HIGH level interrupt, tied LOW
MHU	320	AP_SCP_MHU_0_s_Int	Level	active-HIGH	AP-to-SCP - AP is sender
MHU	321	AP_SCP_MHU_1_s_Int	Level	active-HIGH	AP-to-SCP - AP is sender
MHU	322	AP_SCP_MHU_2_s_Int	Level	active-HIGH	AP-to-SCP - AP is sender
MHU	323	AP_SCP_MHU_3_s_Int	Level	active-HIGH	AP-to-SCP - AP is sender
MHU	324	AP_SCP_MHU_4_s_Int	Level	active-HIGH	AP-to-SCP - AP is sender
MHU	325	AP_SCP_MHU_5_s_Int	Level	active-HIGH	AP-to-SCP - AP is sender
MHU	326	AP_SCP_MHU_6_s_Int	Level	active-HIGH	AP-to-SCP - AP is sender
MHU	327	RESERVED	-	-	-
MHU	328	SCP_AP_MHU_0_r_Int	Level	active-HIGH	SCP-to-AP - AP is receiver
MHU	329	SCP_AP_MHU_1_r_Int	Level	active-HIGH	SCP-to-AP - AP is receiver
MHU	330	SCP_AP_MHU_2_r_Int	Level	active-HIGH	SCP-to-AP - AP is receiver
MHU	331	SCP_AP_MHU_3_r_Int	Level	active-HIGH	SCP-to-AP - AP is receiver
MHU	332	SCP_AP_MHU_4_r_Int	Level	active-HIGH	SCP-to-AP - AP is receiver
MHU	333	SCP_AP_MHU_5_r_Int	Level	active-HIGH	SCP-to-AP - AP is receiver
MHU	334	SCP_AP_MHU_6_r_Int	Level	active-HIGH	SCP-to-AP - AP is receiver

Component	Interrupt id	Interrupt source	Trigger	Polarity	Description
MHU	335	RESERVED	-	-	-
MHU	336	AP_RSS_MHU_0_s_Int	Level	active-HIGH	AP-to-RSE - AP is sender
MHU	337	AP_RSS_MHU_1_s_Int	Level	active-HIGH	AP-to-RSE - AP is sender
MHU	338	AP_RSS_MHU_2_s_Int	Level	active-HIGH	AP-to-RSE - AP is sender
MHU	339	AP_RSS_MHU_3_s_Int	Level	active-HIGH	AP-to-RSE - AP is sender
MHU	340	AP_RSS_MHU_4_s_Int	Level	active-HIGH	AP-to-RSE - AP is sender
MHU	341	RESERVED	-	-	-
MHU	342	RESERVED	-	-	-
MHU	343	RESERVED	-	-	-
MHU	344	RSS_AP_MHU_0_r_Int	Level	active-HIGH	RSE-to-AP - AP is receiver
MHU	345	RSS_AP_MHU_1_r_Int	Level	active-HIGH	RSE-to-AP - AP is receiver
MHU	346	RSS_AP_MHU_2_r_Int	Level	active-HIGH	RSE-to-AP - AP is receiver
MHU	347	RSS_AP_MHU_3_r_Int	Level	active-HIGH	RSE-to-AP - AP is receiver
MHU	348	RSS_AP_MHU_4_r_Int	Level	active-HIGH	RSE-to-AP - AP is receiver
MHU	349	RESERVED	-	-	-
MHU	350	RESERVED	-	-	-
MHU	351	RESERVED	-	-	-
MHU	352-431	RESERVED	Level	active-HIGH	-
NoC	432	Data NoC PMU west_icnclk	Level	active-LOW	-
NoC	433	Data NoC PMU se_icnperiphclk	Level	active-LOW	-
NoC	434	Data NoC PMU gpuclk	Level	active-LOW	-
NoC	435	Data NoC PMU comss_smsclk	Level	active-LOW	-
NoC	436	Data NoC PMU comss_dbgclk	Level	active-LOW	-
NoC	437	Data NoC PMU se_icnclk	Level	active-LOW	-
NoC	438	Data NoC PMU ne_icnclk	Level	active-LOW	-
NoC	439-447	Reserved	-	-	-

Component	Interrupt id	Interrupt source	Trigger	Polarity	Description
NoC	448	Peripheral NoC PMU west_icnclk	Level	active-LOW	-
NoC	449	Peripheral NoC PMU west_icnperiphclk	Level	active-LOW	-
NoC	450	Peripheral NoC PMU gpuclk	Level	active-LOW	-
NoC	451	Peripheral NoC PMU comss_smsclk	Level	active-LOW	-
NoC	452	Peripheral NoC PMU ne_icnperiphclk	Level	active-LOW	-
NoC	453	Peripheral NoC PMU nw_icnperiphclk	Level	active-LOW	-
NoC	454	Peripheral NoC PMU se_icnperiphclk	Level	active-LOW	-
NoC	455	Peripheral NoC PMU sw_icnperiphclk	Level	active-LOW	-
NoC	456	Peripheral NoC PMU comss_dbgclk	Level	active-LOW	-
NoC	457	Peripheral NoC PMU cluster_icnperiphclk	Level	active-LOW	-
NoC	458-463	Reserved	-	-	-
NoC	464	Data PD_SYSTOP_s	Level	active-HIGH	-
NoC	465	Data PD_SYSTOP_ns	Level	active-HIGH	-
NoC	466-479	Reserved	-	-	-
NoC	480	Peripheral PD_SYSTOP_s	Level	active-HIGH	-
NoC	481	Peripheral PD_SYSTOP_ns	Level	active-HIGH	-
NoC	482-511	Reserved	-	-	-

9.2.2 SCP interrupt map

Describes any interrupts that connect to the eWIC of SCP. They are split into several sections according to the hierarchy in the SoC they originate from.


Note

The SCP does not support receiving active-LOW interrupts due to an architecture feature of the V8M cores.

The following tables list the SCP interrupt map:

- SCP core interrupt map

- SCP customization interrupt map
- SCP system integration interrupt map
- SCP integration interrupt map

9.2.2.1 SCP core interrupt map

Describes interrupts required for the underlying IoT subsystems and SCPs are built.

For more information on the SCP core interrupt map, see the *MA1?_ARCHSPEC*.

9.2.2.2 SCP customization interrupt map

Describes the interrupt of the SCP core subsystem representing the complete SCP architecture and only excludes extensions required for the integration into the wider system.

The following table summarizes the SCP customization interrupt map.

Table 9-20: SCP customization interrupt map

Interrupt ID	EWIC Wake support	SCP Architecture defined source	Description	Trigger	Polarity
32-87	-	RESERVED	-	Level	active-HIGH
88	Y	SCP_AP_MHU_0_s_Int	SCP-to-AP - AP is receiver	Level	active-HIGH
89	Y	AP_SCP_MHU_0_r_Int	AP-to-SCP - AP is sender	Level	active-HIGH
90	Y	SCP_AP_MHU_1_s_Int	SCP-to-AP - AP is receiver	Level	active-HIGH
91	Y	AP_SCP_MHU_1_r_Int	AP-to-SCP - AP is sender	Level	active-HIGH
92	Y	SCP_AP_MHU_2_s_Int	SCP-to-AP - AP is receiver	Level	active-HIGH
93	Y	AP_SCP_MHU_2_r_Int	AP-to-SCP - AP is sender	Level	active-HIGH
94	Y	SCP_AP_MHU_3_s_Int	SCP-to-AP - AP is receiver	Level	active-HIGH
95	Y	AP_SCP_MHU_3_r_Int	AP-to-SCP - AP is sender	Level	active-HIGH
96	Y	SCP_AP_MHU_4_s_Int	SCP-to-AP - AP is receiver	Level	active-HIGH
97	Y	AP_SCP_MHU_4_r_Int	AP-to-SCP - AP is sender	Level	active-HIGH
98	Y	SCP_AP_MHU_5_s_Int	SCP-to-AP - AP is receiver	Level	active-HIGH
99	Y	AP_SCP_MHU_5_r_Int	AP-to-SCP - AP is sender	Level	active-HIGH
100	Y	SCP_AP_MHU_6_s_Int	SCP-to-AP - AP is receiver	Level	active-HIGH
101	Y	AP_SCP_MHU_6_r_Int	AP-to-SCP - AP is sender	Level	active-HIGH
102	Y	SCP1_SCP0_MHU_0_r_int	Remote-SCP-to-Local-SCP - Remote-SCP is sender	Level	active-HIGH
103-115	-	RESERVED	-	Level	active-HIGH (tie off `0b0`)
116	Y	ATU INT	Address Translation Unit Interrupt (Secure)	Level	active-HIGH
117-120	-	RESERVED	-	Level	active-HIGH (tie off `0b0`)

9.2.2.3 SCP system integration interrupt map

Describes interrupts of the SCP system integration layer enabling interrupts from peripherals required for the integration of SCP to TCSS-24 platforms.

The following table summarizes the SCP system integration interrupt map.


Note

In Table 9-21: SCP System Integration on page 110, the polarity column shows the interrupt at the source, not the CPU interrupt controller.

Table 9-21: SCP System Integration

Interrupt id	Interrupt source	Wake support	Group	Description	Trigger	Polarity
121	Cluster PPU interrupt	No	PPUs	-	Level	active-HIGH
122	Core 0 PPU interrupt (COREPPUIRQ)	No	PPUs	-	Level	active-HIGH
123	Core 1 PPU interrupt	No	PPUs	-	Level	active-HIGH
124	Core 2 PPU interrupt	No	PPUs	-	Level	active-HIGH
125	Core 3 PPU interrupt	No	PPUs	-	Level	active-HIGH
126	Core 4 PPU interrupt	No	PPUs	-	Level	active-HIGH
127	Core 5 PPU interrupt	No	PPUs	-	Level	active-HIGH
128	Core 6 PPU interrupt	No	PPUs	-	Level	active-HIGH
129	Core 7 PPU interrupt	No	PPUs	-	Level	active-HIGH
130	Core 8 PPU interrupt (CME0PPUIRQ)	No	PPUs	-	Level	active-HIGH
131	Core 9 PPU interrupt (CME1PPUIRQ)	No	PPUs	-	Level	active-HIGH
132-154	RESERVED	No	PPUs	-	-	-
155	SYS PPU0 interrupt	No	PPUs	-	Level	active-HIGH
156	SYS PPU1 interrupt	No	PPUs	-	Level	active-HIGH
157-184	RESERVED	No	PPUs	-	-	-
185-240	RESERVED	No	PLLs	-	-	-

Interrupt id	Interrupt source	Wake support	Group	Description	Trigger	Polarity
241	Cluster Fault/Error IRQ (combined MPAMNSIRQn, MPAMSIRQn CLUSTERCRITIRQn, CLUSTERERRIRQn and CLUSTERFAULTIRQn)	No	CPU	-	Level	active-HIGH
242	Core Fault/Error IRQ (combined COREERRIRQn, COMPLEXERRIRQn, COREFAULTIRQn, COMPLEXFAULTIRQn, CMEFAULTIRQn and CMEERROR_IRQn)	No	CPU	-	Level	active-HIGH
243-248	RESERVED	No	-	-	-	-
249	RESERVED	No	Timer/ Counter	-	-	-
250	SoC (AP) System Timer 0 (S)	Yes	Timer/ Counter	-	Level	active-HIGH
251	SoC (AP) System Timer 1 (NS)	Yes	Timer/ Counter	-	Level	active-HIGH
252	RESERVED	No	Timer/ Counter	-	-	-
253	Trusted SoC (AP) Watchdog (WS0)	No	Timer/ Counter	Bit 0: First Secure WD Expiry (expected to be handled by AP)	Level	active-HIGH
254	Trusted SoC (AP) Watchdog (WS1)	Yes	Timer/ Counter	Bit 1: Second Secure WD Expiry	Level	active-HIGH
255	Generic SoC (AP) Watchdog WS0	No	Timer/ Counter	Bit 0: First Non-secure WD Expiry (expected to be handled by AP)	Level	active-HIGH
256	Generic SoC (AP) Watchdog WS1	Yes	Timer/ Counter	Bit 1: Second Non-secure WD Expiry	Level	active-HIGH
257-280	RESERVED	No	DMC	-	-	-
281	GIC ecc_fatal	No	GIC	SCP should treat this as a request to reset the whole SYSTOP.	-	-
282	GIC axim_err	No	GIC	SCP should treat this as a request to reset the whole SYSTOP.	-	-
283-288	RESERVED	No	GIC	-	-	-
289-305	RESERVED	No	RESERVED	-	-	-
306	RESERVED	Yes	Wake up requests	-	Level	active-HIGH
307-320	RESERVED	Yes	Wake up requests	-	-	-
289	Data PD_SYSTOP_s	No	NoC	-	Level	active-HIGH
290	Data PD_SYSTOP_ns	No	NoC	-	Level	active-HIGH
291-304	Reserved	No	NoC	-	-	
305	Peripheral PD_SYSTOP_s	No	NoC	-	Level	active-HIGH

Interrupt id	Interrupt source	Wake support	Group	Description	Trigger	Polarity
306	Peripheral PD_SYSTOP_ns	No	NoC	-	Level	active-HIGH
307-320	Reserved	No	NoC	-	-	
321	MCN0 - Combined S	No	MCN	irpt_comb_s	Level	active-HIGH
322	MCN0 - Combined NS	No	MCN	irpt_comb_ns	Level	active-HIGH
323	MCN1 - Combined S	No	MCN	-	Level	active-HIGH
324	MCN1 - Combined NS	No	MCN	-	Level	active-HIGH
325	MCN2 - Combined S	No	MCN	-	Level	active-HIGH
326	MCN2 - Combined NS	No	MCN	-	Level	active-HIGH
327	MCN3 - Combined S	No	MCN	-	Level	active-HIGH
328	MCN3 - Combined NS	No	MCN	-	Level	active-HIGH
329	MCN4 - Combined S	No	MCN	-	Level	active-HIGH
330	MCN4 - Combined NS	No	MCN	-	Level	active-HIGH
331	MCN5 - Combined S	No	MCN	-	Level	active-HIGH
332	MCN5 - Combined NS	No	MCN	-	Level	active-HIGH
333	MCN6 - Combined S	No	MCN	-	Level	active-HIGH
334	MCN6 - Combined NS	No	MCN	-	Level	active-HIGH
335	MCN7 - Combined S	No	MCN	-	Level	active-HIGH
336	MCN7 - Combined NS	No	MCN	-	Level	active-HIGH
337	Data NoC PMU west_icnclk	No	NoC	-	Level	active-LOW
338	Data NoC PMU se_icnperiphclk	No	NoC	-	Level	active-LOW
339	Data NoC PMU gpuclk	No	NoC	-	Level	active-LOW
340	Data NoC PMU comss_smsclk	No	NoC	-	Level	active-LOW
341	Data NoC PMU comss_dbgclk	No	NoC	-	Level	active-LOW

Interrupt id	Interrupt source	Wake support	Group	Description	Trigger	Polarity
342	Data NoC PMU se_icnclk	No	NoC	-	Level	active-LOW
343	Data NoC PMU ne_icnclk	No	NoC	-	Level	active-LOW
344-352	Reserved	No	NoC	-	-	
353	Peripheral NoC PMU west_icnclk	No	NoC	-	Level	active-LOW
354	Peripheral NoC PMU west_icnperiphclk	No	NoC	-	Level	active-LOW
355	Peripheral NoC PMU gpuclk	No	NoC	-	Level	active-LOW
356	Peripheral NoC PMU comss_smsclk	No	NoC	-	Level	active-LOW
357	Peripheral NoC PMU ne_icnperiphclk	No	NoC	-	Level	active-LOW
358	Peripheral NoC PMU nw_icnperiphclk	No	NoC	-	Level	active-LOW
359	Peripheral NoC PMU se_icnperiphclk	No	NoC	-	Level	active-LOW
360	Peripheral NoC PMU sw_icnperiphclk	No	NoC	-	Level	active-LOW
361	Peripheral NoC PMU comss_dbgclk	No	NoC	-	Level	active-LOW
362	Peripheral NoC PMU cluster_icnperiphclk	No	NoC	-	Level	active-LOW
363-368	Reserved	No	NoC	-	Level	active-LOW
369	Reserved for SMCF IRQ Collator CPU Block	No	SMCF	-	Level	active-HIGH
370	Reserved for SMCF IRQ Collator GPU Block	No	SMCF	-	Level	active-HIGH
371	Reserved for SMCF IRQ Collator SYS	No	SMCF	-	Level	active-HIGH
372-376	RESERVED	No	SMCF	-	-	-
377	PMIC IRQ_N	Yes	PMIC	-	Level	active-LOW
378	PMIC pre-OCP0 - VBCPU_L	No	PMIC	-	Level	active-LOW
379	PMIC pre-OCP1 - VMCPU_L	No	PMIC	-	Level	active-LOW
380	PMIC pre-OCP2 - VLCPU_L	No	PMIC	-	Level	active-LOW
381	PMIC pre-OCP3 - VGPU_L	No	PMIC	We only support hardware mitigation (clock modulation) for 3 voltage domains. A hardware pre-OCP signal for another domains will have to be exclusively supported in software	Level	active-LOW

Interrupt id	Interrupt source	Wake support	Group	Description	Trigger	Polarity
382-392	RESERVED	No	PMIC	-	-	-

9.2.2.4 SCP integration interrupt map

Describes interrupts of the SCP integration layer enabling interrupts from peripherals required for the integration of SCP into any SoC platforms.

The following table summarizes the SCP integration interrupt map.

Table 9-22: SCP integration

Interrupt id	Interrupt source	Wake support	Description	Trigger	Polarity
393	UART	No	SCP UART Interrupt	Level	active-HIGH
394	GPIO (NS) - combined	Yes	Combined GPIO Interrupt	Level	active-HIGH
395	GPIO (S) - combined	Yes	Combined GPIO Interrupt	Level	active-HIGH
396	GPIO0	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
397	GPIO1	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
398	GPIO2	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
399	GPIO3	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
400	GPIO4	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
401	GPIO5	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
402	GPIO6	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
403	GPIO7	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
404	GPIO8	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
405	GPIO9	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
406	GPIO10	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
407	GPIO11	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
408	GPIO12	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH

Interrupt id	Interrupt source	Wake support	Description	Trigger	Polarity
409	GPIO13	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
410	GPIO14	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
411	GPIO15	Yes	S OR NS interrupt depending on GPIO (S) configuration	Level	active-HIGH
412-413	RESERVED	No	-	-	-
414	Subsystem WAKE	Yes	Wake request from the connected Subsystem ($cpnss for Compute Subsystem SCP)	Level	active-HIGH
415-418	RESERVED	No	-	-	-
419	CS EDAP ROM GPR - cpwrupreq0 (PD_SYSTOP)	Yes	GPR power request for PD_SYSTOP (OR of sys and dbg requests)	Level	active-HIGH
420	CS APP ROM GPR - cpwrupreq0 (PD_CLUSTER)	Yes	GPR power request for PD_CLUSTER (OR of sys and dbg requests)	Level	active-HIGH
421-422	RESERVED	No	-	-	-
423	CS EDAP ROM GPR - csysrstreq	Yes	GPR system reset request	Level	active-HIGH
424-430	RESERVED	No	-	-	-
431	SCP_RSS_MHU_0_s_Int	Yes	-	-	-
432	RSS_SCP_MHU_0_r_Int	Yes	-	-	-
433	CS DAP ROM GPR - cpwrupreq0 (PD_SCP)	Yes	GPR power request for PD_SCP (OR of sys and dbg requests)	Level	active-HIGH
434	CS DAP ROM GPR - cpwrupreq1 (PD_RSS)	Yes	GPR power request for PD_RSS (OR of sys and dbg requests)	Level	active-HIGH
435-441	RESERVED	No	-	-	-
442-479	SCP expansion interrupts, SEI	N/A	This is the SoC expansion block of SCP.	Level	active-HIGH

9.2.3 RSE interrupt map

Describes any interrupts that connect to the eWIC of RSE. They are split into several sections according to the hierarchy in the SoC they originate from.

Note

The RSE does not support receiving active-LOW interrupts due to an architecture feature of the V8M cores.

9.2.3.1 RSE customization

Describes the interrupts required by the RSE core subsystem.

9.2.3.2 RSE integration

Describes the interrupts of the RSE integration layer. It enables interrupts from peripherals required for the integration of RSE into any SoC platform.

The following table summarizes the RSE integration interrupt map.

Table 9-23: RSE integration interrupt map

Interrupt ID	Interrupt Source	Wake support	Description	Trigger	Polarity
96	RSE UART	Yes	UART Combined Error IRQ	Level	active-HIGH
97-101	RESERVED	No	-	Level	Tied low
102	RSS1_RSS0_MHU_0_r_int	Yes	Inter-Subsystem Receiver IRQ	Level	active-HIGH
103	CS EDAP ROM GPR - cpwrupreq0 (PD_SYSTOP)	Yes	Wake up requests. GPR power request for PD_SYSTOP (OR of sys and dbg requests).	Level	active-HIGH
104	CS APP ROM GPR - cpwrupreq0 (PD_CLUSTER)	Yes	Wake up requests. GPR power request for PD_CLUSTER (OR of sys and dbg requests).	Level	active-HIGH
105	CS DAP ROM GPR - cpwrupreq0 (PD_SCP)	Yes	GPR power request for PD_SCP (OR of sys and dbg requests)	Level	active-HIGH
106	CS DAP ROM GPR - cpwrupreq1 (PD_RSS)	Yes	GPR power request for PD_RSS (OR of sys and dbg requests)	Level	active-HIGH
107	CS EDAP ROM GPR - csysrstreq	Yes	GPR system reset request	Level	active-HIGH
108	SDC600 REMRR	Yes	SDC600 Remote Reboot Request	Level	active-HIGH
109-127	RESERVED	No	-	Level	Tied low

9.2.3.3 RSE system integration

Describes the interrupts of the RSE system integration layer enabling peripheral integration specific to the TCSS-24 platform.

The following table summarizes the RSE system integration interrupt map.

Table 9-24: RSE system integration interrupt map

Interrupt ID	Interrupt Source	Wake support	Group	Description	Trigger	Polarity
160	RESERVED	No	Timer/ Counter	-	-	-
161	SoC (AP) System Timer 0 (S)	No	Timer/ Counter	-	Level	active-HIGH
162	SoC (AP) System Timer 1 (NS)	No	Timer/ Counter	-	Level	active-HIGH
163	RESERVED	No	Timer/ Counter	-	-	-

Interrupt ID	Interrupt Source	Wake support	Group	Description	Trigger	Polarity
164	Trusted SoC (AP) Watchdog (WS0)	No	Timer/ Counter	Bit0:FirstSecureWDExpiry	Level	active-HIGH
165	Trusted SoC (AP) Watchdog (WS1)	Yes	Timer/ Counter	Bit1:SecondSecureWDExpiry	Level	active-HIGH
166	Generic SoC (AP) Watchdog WS0	No	Timer/ Counter	Bit0:FirstNon-secureWDExpiry	Level	active-HIGH
167	Generic SoC (AP) Watchdog WS1	Yes	Timer/ Counter	Bit1:SecondNon-secureWDExpiry	Level	active-HIGH
168-191	RESERVED	No	-	-	Level	Tied Low

10. Testing the software stack and FVP

Use confidence tests to verify the functional behavior of the system.


Note

The confidence tests described in this section are not indicative of hardware performance. These tests and the FVP itself are only intended to validate the functional flow and behavior of each of the features.

10.1 Common tests

You can run common tests on either Android or the Buildroot distributions. This section describes the tests common to both distributions and how to run these tests.

10.1.1 Validate pKVM SMMUv3 driver support

You can validate that the SMMUv3 driver is loaded and enabled.

Procedure

1. On the **terminal_uart_ap** enter the following command:

```
realpath /sys/bus/platform/devices/3f000000.iommu/driver
```

   - Example output when the pKVM driver is loaded and successfully enabled:

```
$ realpath /sys/bus/platform/devices/3f000000.iommu/driver
/sys/bus/platform/drivers/kvm-arm-smmu-v3
```

   - Example output when the pKVM driver fails to load or is disabled:

```
$ realpath /sys/bus/platform/devices/3f000000.iommu/driver
/sys/bus/platform/drivers/arm-smmu-v3
```

2. Check the boot log messages for more information about the pKVM driver loading, the initialization phase, and device driver usage. Enter the `dmesg` command for additional information. The `dmesg` command displays entries similar to the following when using hardware rendering:

```
         .
         .
         .
[    0.033341][    T1] iommu: Default domain type: Translated
[    0.033349][    T1] iommu: DMA domain TLB invalidation policy: strict mode
         .
         .
         .
[    0.059858][    T1] kvm [1]: IPA Size Limit: 40 bits
[    0.068132][    T1] kvm-arm-smmu-v3 4002a00000.iommu: ias 40-bit, oas 40-bit
 (features 0x0000dfef)
```

```
[    0.068562][    T1] kvm-arm-smmu-v3 4002a00000.iommu: allocated 65536 entries
 for cmdq
[    0.068574][    T1] kvm-arm-smmu-v3 4002a00000.iommu: 2-level strtab only
 covers 23/32 bits of SID
[    0.070775][    T1] kvm-arm-smmu-v3 3f000000.iommu: ias 40-bit, oas 40-bit
 (features 0x0000dfef)
[    0.071061][    T1] kvm-arm-smmu-v3 3f000000.iommu: allocated 65536 entries
 for cmdq
[    0.071071][    T1] kvm-arm-smmu-v3 3f000000.iommu: 2-level strtab only covers
 23/32 bits of SID
[    0.086915][   T69] Freeing initrd memory: 1428K
[    0.094720][    T1] kvm [1]: GICv4 support disabled
[    0.094727][    T1] kvm [1]: GICv3: no GICV resource entry
[    0.094734][    T1] kvm [1]: disabling GICv2 emulation
[    0.094742][    T1] kvm [1]: GIC system register CPU interface enabled
[    0.094803][    T1] kvm [1]: vgic interrupt IRQ18
[    0.095008][    T1] kvm [1]: Protected nVHE mode initialized successfully
        .
        .
        .
[    0.196354][   T69] komeda 4000000000.display: Adding to iommu group 0
        .
        .
        .
[    3.792147][   T69] mali 2d000000.gpu: Adding to iommu group 1
        .
        .
        .
```

The line `mali 2d000000.gpu: Adding to iommu group 1` in the previous output confirms that the system is using pKVM instead of the classic KVM driver.

10.1.2 Test Trusted Firmware-M

This procedure describes how to run the TF-M test.

Procedure

1. After building your required distribution, run the following commands:

```
export RSE_TESTS=true
./run_docker.sh build-rse.sh all with_reqs
```

2. Run the distribution as normal. The test results display on `rss_terminal_uart`.


Note

The boot process does not fully complete when running the RSE tests.

Results

The following example shows typical TF-M test results:

```
#### Execute test suites for the Secure area ####
Running Test Suite IPC secure interface test (TFM_S_IPC_TEST_1XXX)...
> Executing 'TFM_S_IPC_TEST_1001'
  Description: 'Get PSA framework version'
  TEST: TFM_S_IPC_TEST_1001 - PASSED!
> Executing 'TFM_S_IPC_TEST_1002'
  Description: 'Get version of an RoT Service'
```

```
  TEST: TFM_S_IPC_TEST_1002 - PASSED!
> Executing 'TFM_S_IPC_TEST_1004'
  Description: 'Request connection-based RoT Service'
  TEST: TFM_S_IPC_TEST_1004 - PASSED!
> Executing 'TFM_S_IPC_TEST_1006'
  Description: 'Call PSA RoT access APP RoT memory test service'
Connect success!
Call success!
  TEST: TFM_S_IPC_TEST_1006 - PASSED!
> Executing 'TFM_S_IPC_TEST_1012'
  Description: 'Request stateless service'
  TEST: TFM_S_IPC_TEST_1012 - PASSED!
TESTSUITE PASSED!

.
.
.
TEST: DPE_S_TEST_MUST_BE_THE_LAST - PASSED!
TESTSUITE PASSED!
*** Secure test suites summary ***
Test suite 'IPC secure interface test (TFM_S_IPC_TEST_1XXX)' has PASSED
Test suite 'Crypto secure interface tests (TFM_S_CRYPTO_TEST_1XXX)' has PASSED
Test suite 'Platform Service Secure interface tests(TFM_S_PLATFORM_TEST_1XXX)' has
 PASSED
Test suite 'DPE Secure Tests (DPE_S_TEST_1XXX)' has PASSED
*** End of Secure test suites ***
```

10.1.3 Test the LiteRT ML flow

Validate a typical ML inference flow using the Benchmarking application of the LiteRT Model.

The Model Benchmarking application can consume any LiteRT neural network model file and run a user specified number of inferences on it. This allows you to benchmark performance for the whole graph and for individual operators. See the *TFLite Model Benchmark Tool README* for more information about the Model Benchmarking application.

Required files

This test uses the following files:

`benchmark_model` application

This binary file is built automatically as part of the Lumex build.

`<any model>.tflite` file

You do not need a specific model file but the file you specify must be in valid `.tflite` format. To help you run the confidence test, two model files are provided with the build.

`armNN` folder

This folder contains the files `libarmnn.so`, `libarmnnDelegate.so`, and `Arm_CpuRef_backend.so`. These libraries are required by LiteRT to use ArmNN on the backend to delegate work.

About this task

For Buildroot and Debian distributions, the binaries are automatically integarated into the filesystem, located at `/opt/arm/ml`.

For Android distribution, the binaries are automatically integrated into the filesystem located at `/vendor/opt/arm/ml`.

If you want to use your own LiteRT model, see Manually upload a LiteRT ML model for Buildroot and Debian and Manually upload the required files for Android about uploading your own model to the running Lumex FVP model.

10.1.3.1 Manually upload a LiteRT ML model for Buildroot and Debian

The Buildroot and Debian distributions provide the binaries in the `/opt/arm/ml`. Both distributions also provide the application and the Mobile Object Localizer model in the distribution filesystem. However, you can also upload your own LiteRT model to the running FVP as described in this procedure.

Procedure

1. Download and decompress the MobileNet graph model to your local machine using the following command. This command uses a MobileNet Graph model version as an example, but you can use any valid LiteRT model.

```
# any host path location can be used (as long as it has writable permissions)
mkdir MobileNetGraphTFModel && cd MobileNetGraphTFModel
wget https://storage.googleapis.com/download.tensorflow.org/models/tflite/
mobilenet_v1_224_android_quant_2017_11_08.zip
unzip mobilenet_v1_224_android_quant_2017_11_08.zip
```

2. Upload the MobileNet Graph model to the FVP using the following command:

```
# the following command assumes that the port 8022 is being used as specified in
 the run_model.sh script
scp -P 8022 mobilenet_quant_v1_224.tflite root@localhost:/opt/arm/ml/
# password (if required): root
```

Results

When the model uploads to the remote FVP, you can run the Benchmark Model as described in Run the LiteRT ML model examples.

10.1.3.2 Manually upload the required files for Android

The Android distributions provide the binaries in the `/vendor/opt/arm/ml`. You can also upload your own LiteRT model to the running FVP as described in this procedure.

Procedure

Upload the Mobile Object Localizer model by entering the following commands:

```
cd <TC_OUTPUT>/deploy/
adb connect localhost:5555
adb push mobilenet_quant_v1_224.tflite /data/local/tmp
```

Note

The ADB connection uses the default ADB port 5555. If ADB connection fails or reports that the device is offline, check that you are using the correct ADB port.

Results

When the model uploads to the remote FVP, you can run the Benchmark Model as described in Run the LiteRT ML model examples.

10.1.3.3 Run the LiteRT ML model examples

This procedure describes how to run the benchmark_model application to profile the Mobile Object Localizer LiteRT example model. Although the command arguments vary depending on your use cases and models, the example provides a good starting point for most models.

Procedure

1. In the **terminal_uart_ap** window, log into the FVP running Lumex.
2. Enter the following commands:

```
# the following command ensures correct path location to load the provided
 example ML models
# For Buildroot and Debian distro
   cd /opt/arm/ml
# For Android
   cd /vendor/opt/arm/ml
# With XNNPack for CPU path
  ./benchmark_model --graph=mobile_object_localizer_v1.tflite \
      --num_threads=4 --num_runs=1 --min_secs=0.01 --use_xnnpack=true
# With ArmNN for GPU path (Only available for Android)
  LD_LIBRARY_PATH=/vendor/lib64/egl:armNN/ \
      ./benchmark_model \
      --graph=mobile_object_localizer_v1.tflite \
      --num_threads=4 \
      --num_runs=1 \
      --min_secs=0.01 \
      --external_delegate_path="armNN/libarmnnDelegate.so" \
      --external_delegate_options="backends:GpuAcc;logging-severity:info"
```

Results

The Model Benchmark application runs, profiling the Mobile Object Localizer model. The application then displays statistics and execution information on the terminal window.

10.2 System Monitoring Control Framework

The *System Monitoring Control Framework* (SMCF) is designed to manage a large and diverse set of on-chip sensors and monitors. The SMCF feature can only be used in the Buildroot distribution.

Management of sensors and monitors are done by:

- Presenting software with a standard interface to control the monitors, regardless of type

- Reducing software load of controlling the monitor sampling and data collection

The SMCF reduces the burden on monitor control by enabling sampling on multiple monitors to be controlled together. Sampling control is done through various triggers either internal or external to the SMCF. The number of monitors that the SMCF supports can be configured.

The SMCF also eases data collection requirements by:

- Allowing the data from multiple monitors to be collated in a single location
- Writing out data to a memory-mapped location that is easier for the monitoring agent to access

SMCF can effectively manage sensors, track activity counters, and monitor dynamically evolving system data.

The SMCF consists of two components:

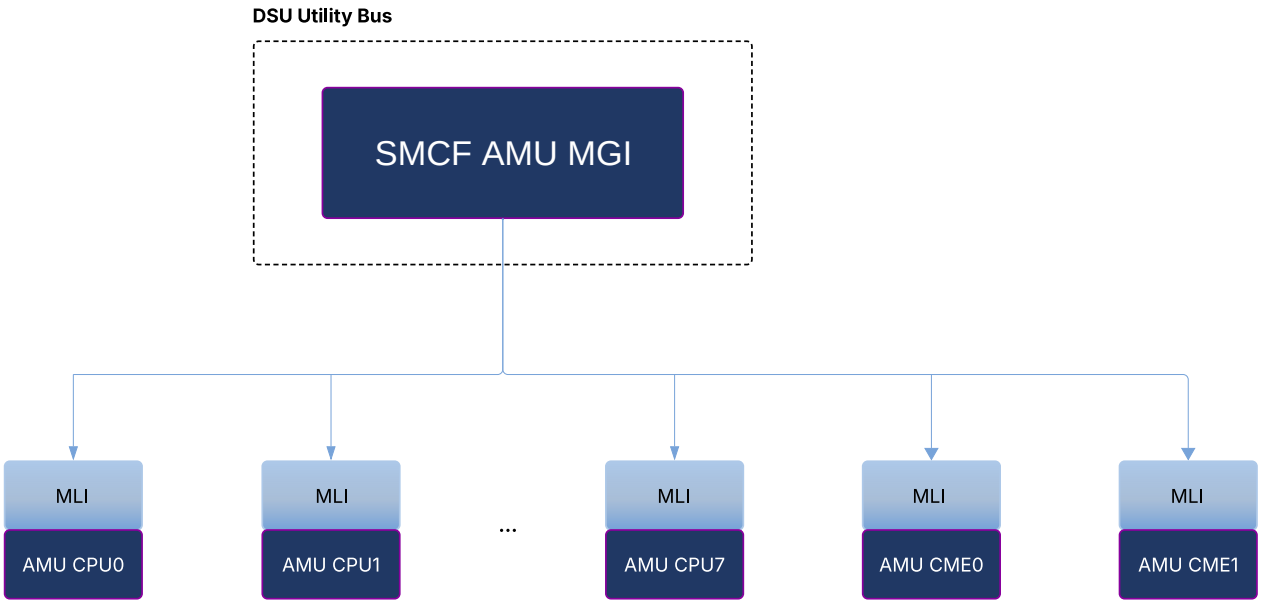
- A Monitor Group Interface (MGI)
- A Monitor Local Interface (MLI)

Each data source is called a monitor and connects to an MLI. The data width of each monitor could be anything from one bit to 64-bits.

Each group of MLI is connected to an MGI, which provides the software interface and a set of functions to be applied to a group of monitors. In Lumex, the MLIs for CME AMUs and CPU AMUs are grouped under one MGI which is implemented in the DSU Utility Bus.

The diagram below shows internal view of the SMCF:

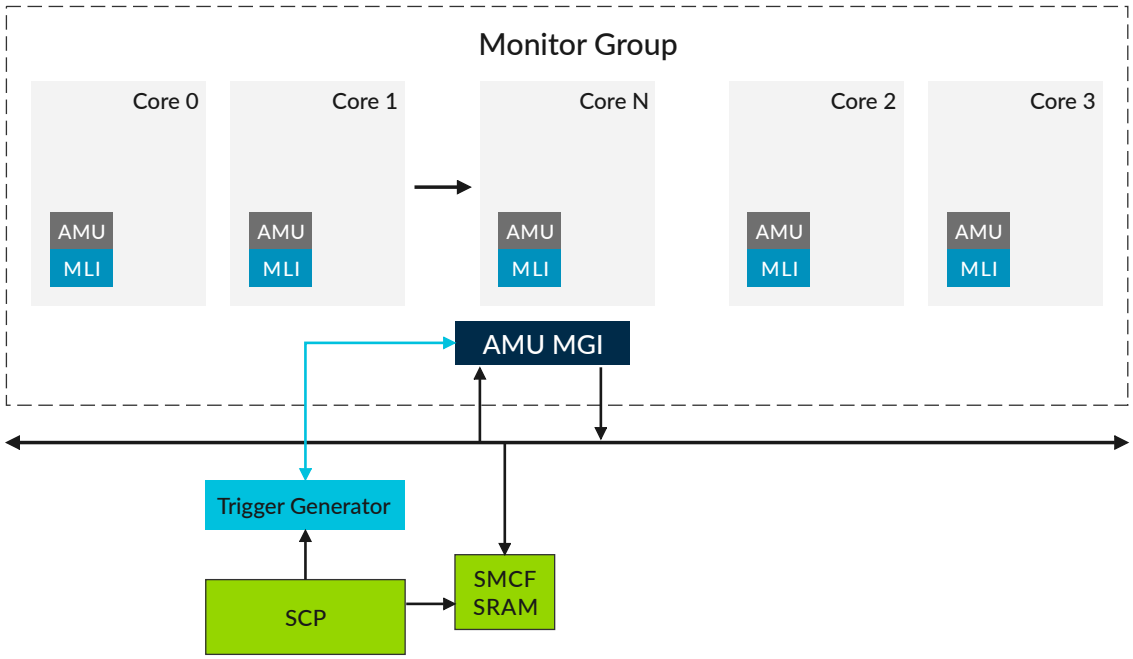
Figure 10-1: SMCF internal view



There is a trigger input from the SCP, which triggers a sample on the SMCF MGI. This allows the SCP to trigger a simultaneous sample on all relevant sensors and monitors.

The diagram below gives the simplified SoC structure of the SMCF:

Figure 10-2: SoC structure of SMCF



There are four modes to sampling the data:

Manual Trigger

Initiated by the software for a single sample from the SMCF

Periodic Sample

Software-driven continuous sampling at predefined interval

Data Read

Data Read sampling is used when a sampling needs to be started after the data from the previous monitor sample data set is consumed. When the last data value from a monitor sample data set is read, a new sample begins.

Input Trigger

External event initiated sampling. Input Trigger sampling is used when a sampling needs to be started from an event that is external to an MGI.

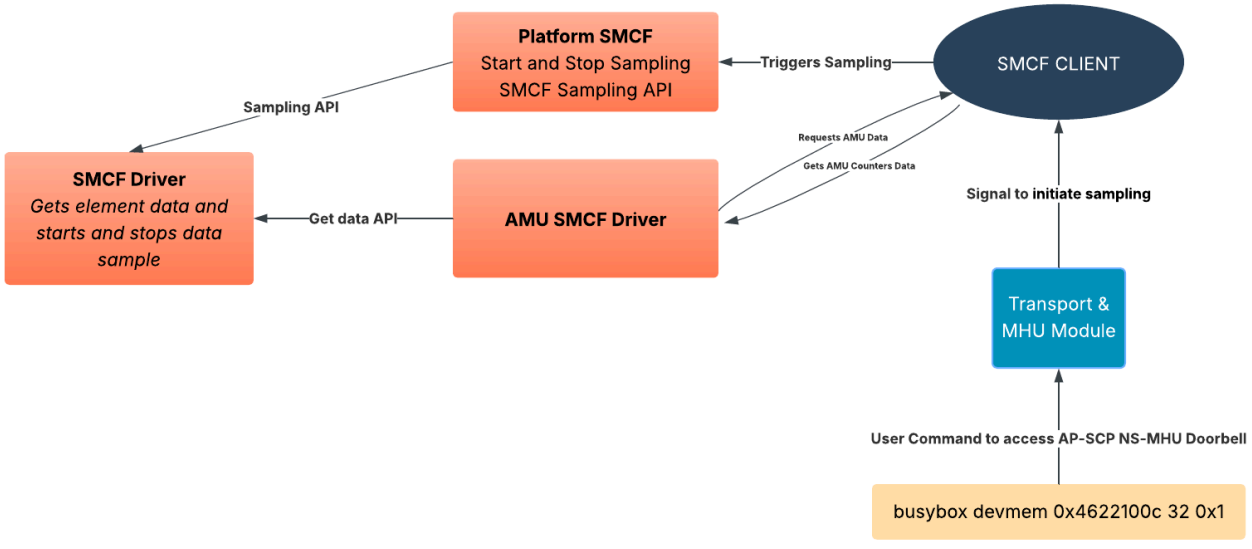
These modes of sampling can be configured via the register MGI_SMP_CFG.SMP_TYP in the MGI. In Lumex, we use Input Trigger sampling for CME AMUs.

SMCF Software Flow and Configuration

1. SCP accesses the SMCF Region through cluster utility map that is mapped to the SCP address translation window.
2. The MGI writes 1 to the Input Sampling trigger `MGI_SMP_EN.EN` to start the sampling process.
3. Software configures the MGI register base address,sample type, MGI write address, SMCF SRAM read address and respective IRQs.
4. Software is expected to write to this SMCF MGI Trigger enable register on a regular interval of time to initiate the sensor data collection. The trigger output from this register is expected to go to all MGIs.
5. The SMCF framework collects the data from MGI and updates the SMCF SRAM on receiving the trigger. Software reads the sensor data from the SMCF SRAM.
6. Any platform with SMCF uses the SMCF to read out the AMU data instead of directly accessing the AMU data.
7. SMCF client module uses AMU smcf and platform smcf module for AMU data collection and for using the data sampling APIs.
8. The platform SMCF module exposes platform specific data sampling APIs, for example start and stop sampling.
9. SMCF client module in SCP binds to AMU SMCF module to read out the AMU data.
10. SMCF client, on receiving instructions from the user, triggers the sampling and gives out AMU data as output in the console.
11. SMCF client is controlled by AP-SCP Non-secure MHU channel. SMCF client binds to Transport module for receiving MHU signal. User from AP Linux console rings AP-SCP Non-secure MHU channel doorbell. On receiving MHU interrupt MHU module through Transport module will signal SMCF client module to start, capture and stop SMCF sampling.

The diagram below explains the software flow of SMCF:

Figure 10-3: Software Flow of SMCF implementation



10.2.1 Validating the SMCF

The SMCF validation uses dummy AMU Counters to generate samples of random numbers. Then it uses the devmem command to access the AP-SCP NS-MHU doorbell channel to read these samples.

Prerequisites

1. Run the `./setup.sh` script
2. `/Dev/mem` is not exposed by default in Buildroot. To expose `/Dev/mem` in Lumex add `CONFIG_DEVMEM=y` to `build-scripts/files/kernel/base.cfg`
3. Enter the `./run_docker.sh ./build-all.sh build` command to build the entire software stack.
4. Pass the following parameters to the FVP through `./run_model.sh <model_params>` to enable dummy AMU Counters and to specify the range of numbers the dummy AMUs pick from:

```
./run-scripts/tc4/run_model.sh -m <path to model> -d buildroot --gpu swr --
-C css.smcf_wrapper.m_dsu_smcf_cluster_CPU_AMU.END_COMPONENT=2
-C css.smcf_wrapper.m_dsu_smcf_cluster_CPU_AMU.FAKE_SENSOR_MIN_LIMIT=256
-C css.smcf_wrapper.m_dsu_smcf_cluster_CPU_AMU.FAKE_SENSOR_MAX_LIMIT=512
```

SMCF sampling procedure

The procedure of SMCF sampling is as follows:

1. Execute devmem command from Linux console for accessing AP-SCP NS-MHU doorbell channel.

```
busybox devmem 0x4622100c 32 0x1
```

2. See output of the sampling.

   Lumex AMUs have 16 counters per AMU, each counter 32 bits wide. The CMEs in Lumex are reprensented under MLI[9] and MLI[10] respectively. Expected output is as follows.

```
[  154.797813] [SMCF_CLIENT] Data successfully fetched for MGI[0] MLI[9]
[  154.797835] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[144] = 0x000d
[  154.797856] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[145] = 0x000b
[  154.797878] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[146] = 0x000f
[  154.797900] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[147] = 0x000a
[  154.797923] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[148] = 0x000f
[  154.797945] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[149] = 0x000d
[  154.797966] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[150] = 0x000b
[  154.797989] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[151] = 0x000c
[  154.798010] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[152] = 0x000f
[  154.798032] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[153] = 0x000f
[  154.798055] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[154] = 0x000d
[  154.798077] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[155] = 0x000b
[  154.798098] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[156] = 0x000b
[  154.798120] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[157] = 0x000a
[  154.798142] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[158] = 0x000d
[  154.798164] [SMCF_CLIENT] MGI[0], MLI[9], MGI_DATA[159] = 0x000e
[  154.798188] [SMCF_CLIENT] Data successfully fetched for MGI[0] MLI[10]
[  154.798210] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[160] = 0x000d
[  154.798231] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[161] = 0x000d
[  154.798254] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[162] = 0x000a
```

```
[  154.798277] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[163] = 0x000b
[  154.798299] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[164] = 0x000c
[  154.798321] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[165] = 0x000e
[  154.798343] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[166] = 0x000f
[  154.798365] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[167] = 0x000a
[  154.798388] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[168] = 0x000a
[  154.798410] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[169] = 0x000f
[  154.798432] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[170] = 0x000b
[  154.798455] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[171] = 0x000b
[  154.798477] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[172] = 0x000d
[  154.798498] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[173] = 0x000a
[  154.798521] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[174] = 0x000e
[  154.798543] [SMCF_CLIENT] MGI[0], MLI[10], MGI_DATA[175] = 0x000b
```

- Pass the following parameters to the FVP through `./run_model.sh <model_params>` to enable to enable dummy AMU Counters:

```
-C css.smcf_wrapper.m_dsu_smcf_cluster_CPU_AMU.END_COMPONENT=2
```

- The dummy AMUs generate samples of random numbers. Apply the following to specify the range of numbers the dummy AMUs pick from:

```
-C css.smcf_wrapper.m_dsu_smcf_cluster_CPU_AMU.FAKE_SENSOR_MIN_LIMIT=<min_limit>
 \
-C css.smcf_wrapper.m_dsu_smcf_cluster_CPU_AMU.FAKE_SENSOR_MAX_LIMIT=<max_limit>
```

- Have a currently running Lumex Buildroot FVP instance. For further instruction on this, see Prepare the compilation environment
- `/Dev/mem` is not exposed by default in Buildroot. Expose `/Dev/mem` in Lumex by setting the following build flag in `build-scripts/files/kernel/base.cfg`:

```
CONFIG_DEVMEM=y
```

10.3 Android tests

You can run Android tests only on the Android distribution. This section describes Android tests and how to run these tests.

10.3.1 Verifying DICE and DPE

The *Device Identifier Composition Engine* (DICE) Layering Scheme is used to help provide attestation for components, and the *DICE Protection Environment* (DPE) stores unique secrets associated with measurements of software components during boot stage.

The Lumex software stack comprises multiple boot components. Enabling Android *protected Virtual Machines* (pVMs) requires the attestation of each component, starting from low-level firmware up to the OS. Lumex provides this attestation using the DICE Layering Scheme to implement a *Boot Certificate Chain* (BCC). Each software component is measured before it is loaded, and a certificate for each boot stage is created using the measurements of the components in that stage. During the

boot stage, the unique secrets associated with each measurement are stored in the DPE. This is a secure partition in the RSE Runtime Firmware. The resulting BCC is verified when the pVM boots.

You can boot an Android pVM as described in Verify the DPE with Microdroid.

10.3.1.1 Verify the DPE with U-boot

You can verify DPE using Android distribution with AVB enabled.

Procedure

1. Build the Android image with AVB enabled if you have not already done so. See Prepare the compilation environment.
2. Run the FVP with the Android distribution and AVB enabled. See Launch the FVP.

Results

On the **terminal_uart_ap**, you see the following output:

```
PVMFW load addr 84000000 size 426 KiB
  Loading PVMFW to f1973000, end f19df5bf ... OK
```

This indicates that the PVMFW image is verified and loaded successfully.

10.3.1.2 Verify the DPE with Microdroid

You can verify the DPE by running Microdroid on Android 14 and Android 15 with AVB enabled.

Procedure

1. Ensure that you export the variables as described in Run Microdroid.
2. Run Microdroid with `--protected` option:

   ```
   ./run-scripts/run_microdroid_demo.sh start-microdroid --protected
   ```

   The script displays output similar to the following:

   ```
   .
   .
   .
   11-25 20:56:54.125    68    68 I vm_payload: vm_payload: Notified host payload
    ready successfully
   11-25 20:56:54.152    69    69 I adbd    : persist.adb.watchdog set to ''
   11-25 20:56:54.152    69    69 I adbd    : persist.sys.test_harness set to ''
   11-25 20:56:54.152    69    69 I adbd    : adb watchdog timeout set to 600
    seconds
   11-25 20:56:54.152    69    69 I adbd    : Setup mdns on port= 5555
   11-25 20:56:54.152    69    72 I adbd    : Waiting for
    persist.adb.tls_server.enable=1
   11-25 20:56:54.153    69    69 I adbd    : adbd listening on vsock:5555
   11-25 20:56:54.153    69    69 I adbd    : adbd started
   .
   .
   .
   ```

3. In a new terminal, with the correct environment variables exported, connect to the pVM. If only one VM instance is running, you do not need to specify the `CID`.

```
./run_microdroid_demo.sh vm-connect <CID>
```

Results

The terminal displays the Microdroid VM shell prompt, meaning that DPE is working correctly. For more information, see Run Microdroid.

10.3.1.3 Verify the BCC

Run the BCC verification test suite to verify that the DPE generated BCC is valid.

Procedure

1. Update the following config variable in `build-scripts/config/common.config`. This enables debug logging for RSE partitions when building the software stack.

```
RSE_PARTITION_LOG_LEVEL="TFM_PARTITION_LOG_LEVEL_DEBUG"
```

2. Build the Android 14 or 15 software stack with AVB enabled.
3. Run the FVP until it fully boots.
4. Once the FVP boots, navigate to `build-scripts/` in a separate terminal and run the following command:

```
./run_docker.sh ../run-scripts/verify_bcc.sh \
 --log latest \
 --platform tc4 \
 --filesystem android \
 --flavor fvp \
 --gpu <your $TC_GPU value>
```

Results

The expected results for the verification test are as follows:

```
Verify Structure: PASS
Certificate Count           : PASS
Component Count (Cert1)     : PASS
Component Count (Cert2)     : PASS
Component Count (Cert3)     : PASS
Component Count (Cert4)     : PASS


Verify Hashes: PASS
RSE_BL1_2           : PASS
RSE_BL2             : PASS
SCP_BL1             : PASS
RSE_S               : PASS
AP_BL1              : PASS
FW_CONFIG           : PASS
TB_FW_CONFIG        : PASS
AP_BL2              : PASS
AP_BL31             : PASS
HW_CONFIG           : PASS
AP_BL32             : PASS
TOS_FW_CONFIG       : PASS
SP_PKG1             : PASS
AP_BL33             : PASS
```

```
NT_FW_CONFIG        : PASS
BOOT                : PASS
PVMFW               : PASS

Verify Issuer Labels: PASS
Cert 1      : PASS
Cert 2      : PASS
Cert 3      : PASS
Cert 4      : PASS
```

This indicates that the PVMFW image is verified and loaded successfully.

10.3.2 Test Trusty IPC

On the Android distribution, Trusty provides a *Trusted Execution Environment* (TEE). This procedure describes how to run the Trusty confidence test.

Procedure

1. To ensure root access, enter `su 0` at the Android boot prompt.
2. Enter the following command:

```
tipc-test -t ta2ta-ipc
```

Results

The following example shows typical test output:

```
console:/ # tipc-test -t ta2ta-ipc
ta2ta_ipc_test:
ipc-unittest-main: 2556: first_free_handle_index: 3
ipc-unittest-main: 2540: retry ret 0, event handle 1000, event 0x1
ipc-unittest-main: 2543: nested ret -13, event handle 1000, event 0x1
[ RUN      ] ipc.wait_negative
[       OK ] ipc.wait_negative
[ RUN      ] ipc.close_handle_negative
[       OK ] ipc.close_handle_negative
[ RUN      ] ipc.set_cookie_negative
[       OK ] ipc.set_cookie_negative
[ RUN      ] ipc.port_create_negative
[       OK ] ipc.port_create_negative
[ RUN      ] ipc.port_create
[       OK ] ipc.port_create
[ RUN      ] ipc.connect_negative
[       OK ] ipc.connect_negative
[ RUN      ] ipc.connect_close
[       OK ] ipc.connect_close
[ RUN      ] ipc.connect_access
[       OK ] ipc.connect_access
[ RUN      ] ipc.accept_negative
[       OK ] ipc.accept_negative
[ DISABLED ] ipc.DISABLED_accept
[ RUN      ] ipc.get_msg_negative
[       OK ] ipc.get_msg_negative
[ RUN      ] ipc.put_msg_negative
[       OK ] ipc.put_msg_negative
[ RUN      ] ipc.send_msg
[       OK ] ipc.send_msg
[ RUN      ] ipc.send_msg_negative
[       OK ] ipc.send_msg_negative
[ RUN      ] ipc.read_msg_negative
[       OK ] ipc.read_msg_negative
```

```
[ RUN      ] ipc.end_to_end_msg
[       OK ] ipc.end_to_end_msg
[ RUN      ] ipc.hset_create
[       OK ] ipc.hset_create
[ RUN      ] ipc.hset_add_mod_del
[       OK ] ipc.hset_add_mod_del
[ RUN      ] ipc.hset_add_self
[       OK ] ipc.hset_add_self
[ RUN      ] ipc.hset_add_loop
[       OK ] ipc.hset_add_loop
[ RUN      ] ipc.hset_add_duplicate
[       OK ] ipc.hset_add_duplicate
[ RUN      ] ipc.hset_wait_on_empty_set
[       OK ] ipc.hset_wait_on_empty_set
[ DISABLED ] ipc.DISABLED_hset_add_chan
[ RUN      ] ipc.send_handle_negative
[       OK ] ipc.send_handle_negative
[ RUN      ] ipc.recv_handle
[       OK ] ipc.recv_handle
[ RUN      ] ipc.recv_handle_negative
[       OK ] ipc.recv_handle_negative
[ RUN      ] ipc.echo_handle_bulk
[       OK ] ipc.echo_handle_bulk
[ RUN      ] ipc.tipc_connect
[       OK ] ipc.tipc_connect
[ RUN      ] ipc.tipc_send_recv_1
[       OK ] ipc.tipc_send_recv_1
[ RUN      ] ipc.tipc_send_recv_hdr_payload
[       OK ] ipc.tipc_send_recv_hdr_payload
[==========] 28 tests ran.
[  PASSED  ] 28 tests.
[ DISABLED ] 2 tests.
console:/ #
```

10.3.3 Run Microdroid

On the Android distribution, the Virtualization service provides support to run the Microdroid-based *Protected Virtual Machine* (pVM). This section describes how to run either the Microdroid demo or an actual Microdroid instance.

Before you begin

1. Boot the Lumex FVP with the Android distribution. After the Android homescreen renders, wait approximately 30 minutes to ensure the ADB service starts.
2. From the host console terminal, run the following commands:

```
export TC_ANDROID_VERSION=android15
export ANDROID_PRODUCT_OUT=(...)/product/tc_fvp_swr/
```

Procedure

1. To run the Microdroid demo, enter the following command, otherwise skip to the next step:

   ```
   ./run-scripts/run_microdroid_demo.sh run-tc-app
   ```

   The demo produces the test results listed in the Results section.
2. To run a Microdroid instance, enter the following command:

   ```
   ./run-scripts/run_microdroid_demo.sh start-microdroid
   ```

The terminal waits for an ADB connection.

3. To connect to a Microdroid instance with ADB, do one of the following depending on the number of instances:

   - For one Microdroid instance, enter the following command when the Microdroid instance starts:

     ```
     ./run-scripts/run_microdroid_demo.sh start-microdroid --auto-connect
     ```

   - For multiple Microdroid instances, start the Microdroid instances, then enter the following command on a different console terminal:

     ```
     ./run_microdroid_demo.sh vm-connect <CID>
     ```

     The CID for the Microdroid instance is displayed when the instance starts. In addition, the script prompts you to select between the running instances.


Note

This test is specific to Android. The ADB connection uses the default ADB port 5555. If ADB connection fails or reports that the device is offline, check the ADB port in use and edit the script accordingly.

Results

The Microdroid demo test produces results similar to the following:

```
        .
        .
        .
INFO: APK was built successfully.
INFO: ADB connecting to 127.0.0.1:5555
INFO: ADB connected to 127.0.0.1:5555
INFO: Checking ro.product.name
INFO: ro.product.name matches tc_fvp
INFO: Checking path of com.android.microdroid.tc
INFO: APK Installed path is: /system/app/TCMicrodroidDemoApp/TCMicrodroidDemoApp.apk
Created VM from "/data/local/tmp/virt/TCMicrodroidDemoApp.apk"!ConfigPath("assets/
vm_config.json") with CID 2049, state is STARTING.
[2024-04-03T09:17:23.132825560+00:00 INFO  crosvm] crosvm started.
[2024-04-03T09:17:23.133432232+00:00 INFO  crosvm] CLI arguments parsed.
[2024-04-03T09:17:23.149928488+00:00
 INFO  crosvm::crosvm::sys::unix::device_helpers] Trying to attach block device: /
proc/self/fd/49
[2024-04-03T09:17:23.150215128+00:00
 INFO  crosvm::crosvm::sys::unix::device_helpers] Trying to attach block device: /
proc/self/fd/54
[2024-04-03T09:17:23.196606144+00:00
 INFO  crosvm::crosvm::sys::unix::device_helpers] Trying to attach block device: /
proc/self/fd/63
[    0.089283][   T21] Freeing initrd memory: 1652K
[    0.089863][   T17] cacheinfo: Unable to detect cache hierarchy for CPU 0
[    0.091308][    T1] brd: module loaded
[    0.093654][    T1] loop: module loaded
[    0.093768][    T1] virtio_blk virtio3: 1/0/0 default/read/poll queues
        .
        .
        .
```

```
[    0.311394][    T1] init: Loading SELinux policy
[    0.312782][    T1] SELinux:  Permission nlmsg_getneigh in class
 netlink_route_socket not defined in policy.
[    0.312995][    T1] SELinux:  Permission bpf in class capability2 not defined in
 policy.
[    0.313093][    T1] SELinux:  Permission checkpoint_restore in class capability2
 not defined in policy.
[    0.313238][    T1] SELinux:  Permission bpf in class cap2_userns not defined in
 policy.
[    0.313355][    T1] SELinux:  Permission checkpoint_restore in class cap2_userns
 not defined in policy.
[    0.313555][    T1] SELinux:  Class mctp_socket not defined in policy.
[    0.313645][    T1] SELinux:  Class io_uring not defined in policy.
[    0.313756][    T1] SELinux:  Class user_namespace not defined in policy.
[    0.313859][    T1] SELinux: the above unknown classes and permissions will be
 denied
[    0.318081][    T1] SELinux:  policy capability network_peer_controls=1
        .
        .
        .
[    0.394310][    T1] selinux: SELinux: Loaded file context from:
[    0.394524][    T1] selinux:             /system/etc/selinux/plat_file_contexts
[    0.396143][    T1] init: Using Android DT directory /proc/device-tree/firmware/
android/
[    0.397222][    T1] init: Setting property 'ro.build.fingerprint' to 'unknown/
unknown/unknown:unknown/.4349bf74/unknown:unknown/unknown'
[    0.397930][    T1] selinux: SELinux: Loaded file context from:
[    0.398164][    T1] selinux:             /system/etc/selinux/plat_file_contexts
[    0.398375][    T1] init: Running restorecon...
[    0.400021][    T1] selinux: SELinux: Could not get canonical path for /metadata/
ota/rollback-indicator restorecon: No such file or directory.
[    0.400442][    T1] selinux: SELinux: Could not get canonical path for /metadata/
gsi restorecon: No such file or directory.
[    0.400845][    T1] init: Created socket '/dev/socket/property_service', mode
 666, user 0, group 0
[    0.401388][    T1] init: [libfs_mgr] vendor overlay: vndk version not defined
[    0.401890][    T1] init: SetupMountNamespaces done
[    0.402051][    T1] init: Not using subcontext for microdroid
[    0.402506][    T1] init: Parsing file /system/etc/init/hw/init.rc...
[    0.402947][    T1] init: Added '/init.environ.rc' to import list
[    0.403334][    T1] init: Parsing file /init.environ.rc...
[    0.403500][    T1] init: Unable to read config file '/init.environ.rc': open()
 failed: No such file or directory
        .
        .
        .
[    0.487280][    T1] init: starting service 'microdroid_manager'...
[    0.487467][    T1] init: Created socket '/dev/socket/vm_payload_service', mode
 666, user 1000, group 1000
[    0.488369][    T1] init: ... started service 'microdroid_manager' has pid 54
[    0.488570][   T49] ueventd: Coldboot took 0.041 seconds
        .
        .
        .
[    0.550323][   T54] microdroid_manager[54]: started.
[    0.550495][   T54] microdroid_manager[54]: ramdump supported: true
[    0.550615][   T54] microdroid_manager[54]: Os { code: 2, kind: NotFound,
 message: "No such file or directory" }. Assumes <0>
        .
        .
        .
[    0.643333][   T54] microdroid_manager[54]: ramdump is loaded: debuggable=true,
 ramdump=false
[    0.643757][   T54] zram0: detected capacity change from 0 to 454240
[    0.643944][   T54] Adding 227116k swap on /dev/block/zram0.  Priority:-2
 extents:1 across:227116k SS
[    0.644101][   T54] microdroid_manager[54]: swap enabled.
[    0.644248][   T55] kexec_load (55) used greatest stack depth: 11968 bytes left
[    0.645184][   T54] microdroid_manager::payload[54]: loading payload metadata...
[    0.645326][   T54] microdroid_manager::ioutil[54]: waiting for "/dev/block/by-
name/payload-metadata"...
```

```
[    0.645526][   T54] microdroid_manager::dice[54]: Using sample DICE values
        .
        .
        .
[    0.691145][   T54] microdroid_manager[54]: payload verification successful. took
 35.677936ms
[    0.691307][   T54] microdroid_manager[54]: Saved data is verified.
[    0.691435][   T54] microdroid_manager[54]: DICE derivation for payload
[    0.698650][   T54] microdroid_manager[54]: loading config from "/mnt/apk/assets/
vm_config.json"...
[    0.698819][   T54] microdroid_manager::ioutil[54]: waiting for "/mnt/apk/assets/
vm_config.json"...
        .
        .
        .
[    0.809250][   T54] microdroid_manager::vm_payload_service[54]: The RPC server
 'vm_payload_service' is running.
        .
        .
        .
[    0.817050][   T54] microdroid_manager[54]: boot completed, time to run payload
[    0.817176][   T54] microdroid_manager[54]: executing main task Task { type_:
 MicrodroidLauncher, command: "TCMicrodroidApp.so" }...
[    0.817391][   T54] microdroid_manager[54]: notifying payload started
[    0.817551][    T1] init: processing action (enable_property_trigger) from
 (<Builtin Action>:0)
payload started
[    0.818000][    T1] init: processing action (microdroid_manager.init_done=1) from
 (/system/etc/init/hw/init.rc:49)
[    0.818196][    T1] init: Sending signal 9 to service 'ueventd' (pid 49) process
 group...
[    0.818872][    T1] libprocessgroup: Successfully killed process cgroup uid 0 pid
 49 in 0ms
[    0.819032][    T1] init: Service 'ueventd' (pid 49) received signal 9
[    0.819238][    T1] init: processing action (init_debug_policy.adbd.enabled=1)
 from (/system/etc/init/hw/init.rc:57)
[    0.819539][    T1] init: starting service 'adbd'...
[    0.819637][    T1] init: Created socket '/dev/socket/adbd', mode 660, user 1000,
 group 1000
[    0.820284][    T1] init: ... started service 'adbd' has pid 74
[    0.820686][   T73] microdroid_manager[73]: dropping capabilities before
 executing payload
[    0.893050][   T73] microdroid_launcher: Hello Microdroid!
[    0.893130][   T73] microdroid_launcher:
[    0.893189][   T73] microdroid_launcher:
[    0.893552][   T54] microdroid_manager[54]: task successfully finished
[    0.893648][   T54] microdroid_manager[54]: notifying payload finished
payload finished with exit code 0
[    0.893841][   T54] microdroid_manager[54]: Shutting down...
[    0.894009][   T48] init: Received sys.powerctl='shutdown' from pid: 54 (/system/
bin/microdroid_manager)
[    0.894193][    T1] init: Got shutdown_command 'shutdown' Calling
 HandlePowerctlMessage()
[    0.894317][    T1] init: Clear action queue and start shutdown trigger
[    0.894430][    T1] init: Entering shutdown mode
        .
        .
        .
[2024-04-03T09:17:24.547673048+00:00 INFO  crosvm] exiting with success
VM ended: Shutdown
        .
        .
        .
```

10.3.4 Test BTI

You can use *Branch Target Identification* (BTI) to guard against the execution of instructions that are not the intended target of a branch. This procedure describes how to run the BTI confidence test.

Procedure

On the **terminal_uart_ap** terminal, enter the following commands:

```
su
cd /data/nativetest64/bti-unit-tests/
./bti-unit-tests
```

Results

The following example shows typical test output:

```
console:/data/nativetest64/bti-unit-tests # ./bti-unit-tests

[==========] Running 17 tests from 7 test suites.
[----------] Global test environment set-up.
[----------] 3 tests from BR_Test
[ RUN      ] BR_Test.GuardedMemoryWithX16OrX17
[       OK ] BR_Test.GuardedMemoryWithX16OrX17 (206 ms)
[ RUN      ] BR_Test.NonGuardedMemoryAnyRegister
[       OK ] BR_Test.NonGuardedMemoryAnyRegister (0 ms)
[ RUN      ] BR_Test.GuardedMemoryOtherRegisters
[       OK ] BR_Test.GuardedMemoryOtherRegisters (155 ms)
[----------] 3 tests from BR_Test (362 ms total)

[----------] 3 tests from BRAA_Test
[ RUN      ] BRAA_Test.GuardedMemoryWithX16OrX17
[       OK ] BRAA_Test.GuardedMemoryWithX16OrX17 (429 ms)
[ RUN      ] BRAA_Test.NonGuardedMemoryAnyRegister
[       OK ] BRAA_Test.NonGuardedMemoryAnyRegister (0 ms)
[ RUN      ] BRAA_Test.GuardedMemoryOtherRegisters
[       OK ] BRAA_Test.GuardedMemoryOtherRegisters (283 ms)
[----------] 3 tests from BRAA_Test (713 ms total)

[----------] 3 tests from BRAB_Test
[ RUN      ] BRAB_Test.GuardedMemoryWithX16OrX17
[       OK ] BRAB_Test.GuardedMemoryWithX16OrX17 (385 ms)
[ RUN      ] BRAB_Test.NonGuardedMemoryAnyRegister
[       OK ] BRAB_Test.NonGuardedMemoryAnyRegister (0 ms)
[ RUN      ] BRAB_Test.GuardedMemoryOtherRegisters
[       OK ] BRAB_Test.GuardedMemoryOtherRegisters (297 ms)
[----------] 3 tests from BRAB_Test (682 ms total)

[----------] 2 tests from BLR_Test
[ RUN      ] BLR_Test.GuardedMemoryAnyRegister
[       OK ] BLR_Test.GuardedMemoryAnyRegister (427 ms)
[ RUN      ] BLR_Test.NonGuardedMemoryAnyRegister
[       OK ] BLR_Test.NonGuardedMemoryAnyRegister (0 ms)
[----------] 2 tests from BLR_Test (427 ms total)

[----------] 2 tests from BLRAA_Test
[ RUN      ] BLRAA_Test.GuardedMemoryAnyRegister
[       OK ] BLRAA_Test.GuardedMemoryAnyRegister (936 ms)
[ RUN      ] BLRAA_Test.NonGuardedMemoryAnyRegister
[       OK ] BLRAA_Test.NonGuardedMemoryAnyRegister (0 ms)
[----------] 2 tests from BLRAA_Test (937 ms total)

[----------] 2 tests from BLRAB_Test
[ RUN      ] BLRAB_Test.GuardedMemoryAnyRegister
[       OK ] BLRAB_Test.GuardedMemoryAnyRegister (749 ms)
[ RUN      ] BLRAB_Test.NonGuardedMemoryAnyRegister
[       OK ] BLRAB_Test.NonGuardedMemoryAnyRegister (0 ms)
```

```
[----------] 2 tests from BLRAB_Test (749 ms total)

[----------] 2 tests from BTI_LinkerTest
[ RUN      ] BTI_LinkerTest.CallBasicFunction
[       OK ] BTI_LinkerTest.CallBasicFunction (0 ms)
[ RUN      ] BTI_LinkerTest.BypassLandingPad
[       OK ] BTI_LinkerTest.BypassLandingPad (55 ms)
[----------] 2 tests from BTI_LinkerTest (55 ms total)

[----------] Global test environment tear-down
[==========] 17 tests from 7 test suites ran. (3929 ms total)
[  PASSED  ] 17 tests.
```

10.3.5 Test the MTE

Arm *Memory Tagging Extension* (MTE) is a hardware extension that allows you to catch use-after-free and buffer-overflow bugs in your native code. This procedure describes how to run the MTE confidence test.

Procedure

On the **terminal_uart_ap** terminal enter the following commands:

```
su
cd /data/nativetest64/mte-unit-tests/
./mte-unit-tests
```

Results

The following example shows typical test output:

```
console:/data/nativetest64/mte-unit-tests # ./mte-unit-tests

[==========] Running 12 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 12 tests from MTETest
[ RUN      ] MTETest.CreateRandomTag
[       OK ] MTETest.CreateRandomTag (0 ms)
[ RUN      ] MTETest.IncrementTag
[       OK ] MTETest.IncrementTag (0 ms)
[ RUN      ] MTETest.ExcludedTags
[       OK ] MTETest.ExcludedTags (0 ms)
[ RUN      ] MTETest.PointerSubtraction
[       OK ] MTETest.PointerSubtraction (0 ms)
[ RUN      ] MTETest.TagStoreAndLoad
[       OK ] MTETest.TagStoreAndLoad (0 ms)
[ RUN      ] MTETest.DCGZVA
[       OK ] MTETest.DCGZVA (0 ms)
[ RUN      ] MTETest.DCGVA
[       OK ] MTETest.DCGVA (0 ms)
[ RUN      ] MTETest.Segfault
[       OK ] MTETest.Segfault (41 ms)
[ RUN      ] MTETest.UseAfterFree
[       OK ] MTETest.UseAfterFree (0 ms)
[ RUN      ] MTETest.CopyOnWrite
[       OK ] MTETest.CopyOnWrite (0 ms)
[ RUN      ] MTETest.mmapTempfile
[       OK ] MTETest.mmapTempfile (5 ms)
[ RUN      ] MTETest.MTEIsEnabled
[       OK ] MTETest.MTEIsEnabled (0 ms)
[----------] 12 tests from MTETest (48 ms total)

[----------] Global test environment tear-down
[==========] 12 tests from 1 test suite ran. (48 ms total)
```

```
[  PASSED  ] 12 tests.
```

10.3.6 Test the PAUTH

Pointer AUTHentication (PAUTH) is designed to mitigate *Return-Oriented Programming* (ROP) security exploit attacks.

Procedure

Enter the following command in the **terminal_uart_ap** window:

```
su
cd /data/nativetest64/pauth-unit-tests/
./pauth-unit-tests
```

Results

The following example shows typical test output:

```
console:/data/nativetest64/pauth-unit-tests $ ./pauth-unit-tests
PAC is enabled by the kernel: 1
PAC2 is implemented by the hardware: 1
FPAC is implemented by the hardware: 1
[==========] Running 18 tests from 3 test suites.
[----------] Global test environment set-up.
[----------] 2 tests from PAuthDeathTest
[ RUN      ] PAuthDeathTest.SignFailure
[       OK ] PAuthDeathTest.SignFailure (113 ms)
[ RUN      ] PAuthDeathTest.AuthFailure
[       OK ] PAuthDeathTest.AuthFailure (137 ms)
[----------] 2 tests from PAuthDeathTest (250 ms total)

[----------] 13 tests from PAuthTest
[ RUN      ] PAuthTest.Signing
[       OK ] PAuthTest.Signing (0 ms)
[ RUN      ] PAuthTest.Authentication
[       OK ] PAuthTest.Authentication (146 ms)
[ RUN      ] PAuthTest.Stripping
vendor/arm/examples/pauth/pauth_unit_tests/pauth_unit_tests.cpp:279: Skipped

[  SKIPPED ] PAuthTest.Stripping (0 ms)
[ RUN      ] PAuthTest.Roundtrip
[       OK ] PAuthTest.Roundtrip (0 ms)
[ RUN      ] PAuthTest.StrippingWithBuiltinReturnAddress
[       OK ] PAuthTest.StrippingWithBuiltinReturnAddress (0 ms)
[ RUN      ] PAuthTest.ExtractPAC
[       OK ] PAuthTest.ExtractPAC (0 ms)
[ RUN      ] PAuthTest.PACMask
[       OK ] PAuthTest.PACMask (0 ms)
[ RUN      ] PAuthTest.KeyChange
[       OK ] PAuthTest.KeyChange (1 ms)
[ RUN      ] PAuthTest.GenericAuthentication
[       OK ] PAuthTest.GenericAuthentication (0 ms)
[ RUN      ] PAuthTest.Unwind
[       OK ] PAuthTest.Unwind (8 ms)
[ RUN      ] PAuthTest.CheckReturnAddressSigned
[       OK ] PAuthTest.CheckReturnAddressSigned (0 ms)
[ RUN      ] PAuthTest.AuthenticateThenReturn
[       OK ] PAuthTest.AuthenticateThenReturn (93 ms)
[ RUN      ] PAuthTest.CheckHWCAP
[       OK ] PAuthTest.CheckHWCAP (0 ms)
[----------] 13 tests from PAuthTest (251 ms total)

[----------] 3 tests from PAuthTestData
```

```
[ RUN      ] PAuthTestData.Signing
[       OK ] PAuthTestData.Signing (0 ms)
[ RUN      ] PAuthTestData.Authentication
[       OK ] PAuthTestData.Authentication (92 ms)
[ RUN      ] PAuthTestData.Roundtrip
[       OK ] PAuthTestData.Roundtrip (0 ms)
[----------] 3 tests from PAuthTestData (92 ms total)

[----------] Global test environment tear-down
[==========] 18 tests from 3 test suites ran. (594 ms total)
[  PASSED  ] 17 tests.
[  SKIPPED ] 1 test, listed below:
[  SKIPPED ] PAuthTest.Stripping
```

10.4 Buildroot tests

You can run Buildroot tests only on the Buildroot distribution. This section describes Buildroot tests and how to run these tests.

10.4.1 Test the OP-TEE

OP-TEE is a *Trusted Execution Environment* (TEE). This procedure describes how to run the OP-TEE confidence test.

Procedure

Run the test suite by entering the `xtest` command on the **terminal_uart_ap** terminal.


Note

Running the entire test suite takes approximately 30 minutes.

Results

The test suite produces results similar to the following:

```
# xtest
Run test suite with level=0

TEE test application started over default TEE instance
##################################################
#
# regression
#
##################################################

* regression_1001 Core self tests
 - 1001 -   skip test, pseudo TA not found
  regression_1001 OK

* regression_1002 PTA parameters
 - 1002 -   skip test, pseudo TA not found
  regression_1002 OK
        .
        .
```

```
.
regression_8101 OK
regression_8102 OK
regression_8103 OK
+-----------------------------------------------------
29678 subtests of which 0 failed
107 test cases of which 0 failed
0 test cases were skipped
TEE test application done!
#
```

10.4.2 Test Trusted Services and client application

You can run the Trusted Service tests and client application demo on Buildroot.

Procedure

1. To test Trusted Services at the service API-level, enter the following command:

```
ts-service-test -g FwuServiceTests -g ItsServiceTests -g
 CryptoKeyDerivationServicePackedcTests -g CryptoMacServicePackedcTests
 -g CryptoCipherServicePackedcTests -g CryptoHashServicePackedcTests
 -g CryptoServicePackedcTests -g CryptoServiceProtobufTests -g
 CryptoServiceLimitTests -v
```

2. Run `ts-demo` for a demonstration of the client application.

Results

The following example shows typical Trusted Services test results:

```
# ts-service-test -g FwuServiceTests -g ItsServiceTests -g
 CryptoKeyDerivationServicePackedcTests -g CryptoMacServicePackedcTests
 -g CryptoCipherServicePackedcTests -g CryptoHashServicePackedcTests -g
 CryptoServicePackedcTests -g CryptoServiceProtobufTests -g CryptoServiceLimitTests
 -v
TEST(FwuServiceTests, checkMetadataAccess) - 962 ms
TEST(FwuServiceTests, checkImgDirAccess) - 894 ms
TEST(ItsServiceTests, storeNewItem) - 977 ms
TEST(CryptoKeyDerivationServicePackedcTests, deriveAbort) - 908 ms
TEST(CryptoKeyDerivationServicePackedcTests, hkdfDeriveBytes) - 968 ms
TEST(CryptoKeyDerivationServicePackedcTests, hkdfDeriveKey) - 949 ms
TEST(CryptoMacServicePackedcTests, macAbort) - 891 ms
TEST(CryptoMacServicePackedcTests, signAndVerify) - 3208 ms
TEST(CryptoCipherServicePackedcTests, cipherAbort) - 896 ms
TEST(CryptoCipherServicePackedcTests, encryptDecryptRoundtrip) - 1715 ms
TEST(CryptoHashServicePackedcTests, hashAbort) - 713 ms
TEST(CryptoHashServicePackedcTests, hashAndVerify) - 858 ms
TEST(CryptoHashServicePackedcTests, calculateHash) - 641 ms
TEST(CryptoServicePackedcTests, getUefiPrivAuthVarFingerprint) - 666 ms
TEST(CryptoServicePackedcTests, verifyPkcs7Signature) - 5661 ms
TEST(CryptoServicePackedcTests, generateRandomNumbers) - 700 ms
TEST(CryptoServicePackedcTests, asymEncryptDecryptWithSalt) - 73067 ms
TEST(CryptoServicePackedcTests, asymEncryptDecrypt) - 62412 ms
TEST(CryptoServicePackedcTests, signAndVerifyEat) - 12352 ms
TEST(CryptoServicePackedcTests, signAndVerifyMessage) - 12439 ms
TEST(CryptoServicePackedcTests, signAndVerifyHash) - 12465 ms
TEST(CryptoServicePackedcTests, exportAndImportKeyPair) - 1646 ms
TEST(CryptoServicePackedcTests, exportPublicKey) - 2582 ms
TEST(CryptoServicePackedcTests, purgeKey) - 1584 ms
TEST(CryptoServicePackedcTests, copyKey) - 87618 ms
TEST(CryptoServicePackedcTests, generatePersistentKeys) - 2622 ms
TEST(CryptoServicePackedcTests, generateVolatileKeys) - 2504 ms
TEST(CryptoServiceProtobufTests, generateRandomNumbers) - 621 ms
TEST(CryptoServiceProtobufTests, asymEncryptDecryptWithSalt) - 38097 ms
```

```
TEST(CryptoServiceProtobufTests, asymEncryptDecrypt) - 55196 ms
TEST(CryptoServiceProtobufTests, signAndVerifyMessage) - 12452 ms
TEST(CryptoServiceProtobufTests, signAndVerifyHash) - 12486 ms
TEST(CryptoServiceProtobufTests, exportAndImportKeyPair) - 1662 ms
TEST(CryptoServiceProtobufTests, exportPublicKey) - 2558 ms
TEST(CryptoServiceProtobufTests, generatePersistentKeys) - 2640 ms
TEST(CryptoServiceProtobufTests, generateVolatileKeys) - 2525 ms
TEST(CryptoServiceLimitTests, volatileRsaKeyPairLimit) - 2101315 ms
TEST(CryptoServiceLimitTests, volatileEccKeyPairLimit) - 84459 ms

OK (46 tests, 38 ran, 328 checks, 0 ignored, 8 filtered out, 2607650 ms)

#
```

The following example shows a typical run of the client application demo:

```
# ts-demo

Demonstrates use of trusted services from an application
--------------------------------------------------------
A client requests a set of crypto operations performed by
the Crypto service.  Key storage for persistent keys is
provided by the Secure Storage service via the ITS client.
Generating random bytes length: 1
        Operation successful
        Random bytes:
                EB
Generating random bytes length: 7
        Operation successful
        Random bytes:
                10 1A D4 42 3C 77 37
Generating random bytes length: 128
        Operation successful
        Random bytes:
                30 44 6A 8E 29 69 09 94
                1D F4 51 01 0D E3 95 64
                D4 C4 41 B4 5B 3C E9 10
                FC 03 1E 4C 84 D3 3D 95
                93 1C EA 44 F1 3B 74 2F
                39 25 AC 69 E9 03 12 1C
                E0 07 D2 B6 1E 8E E6 1D
                AE 33 77 28 A4 F7 1E C6
                BA FE 56 BF D2 DF 7A A3
                2C FE C0 80 8F 37 5F A9
                4D 5A 7D C7 25 E1 92 67
                29 CB 77 F6 B5 E4 D8 11
                12 E4 47 5E 59 94 D9 0C
                9E A6 57 44 7C F3 6D BE
                E2 A0 08 53 48 B8 F8 EC
                D2 6D F3 FA 79 10 C3 6D
Generating ECC signing key
        Operation successful
Signing message: "The quick brown fox" using key: 256
        Operation successful
        Signature bytes:
                68 0A 10 55 09 22 1E 3D
                0F 1D E8 07 C1 8D 81 52
                8C D5 10 6D 11 63 1E 73
                BE A8 F5 EB D9 51 2D 13
                F6 8B 89 98 77 6D 49 17
                B8 A3 4D 0E 1D F9 98 49
                5C 6E 49 EC 16 6E 9F C3
                63 C5 16 95 EA 46 48 5D
Verify signature using original message: "The quick brown fox"
        Operation successful
Verify signature using modified message: "!he quick brown fox"
        Successfully detected modified message
Signing message: "jumps over the lazy dog" using key: 256
```

```
        Operation successful
        Signature bytes:
                C4 7C 44 17 9F 36 CD 82
                4E 98 DA B2 F5 1D 59 D6
                AC 74 5B E7 E7 92 DE 1E
                71 EF 54 C8 E0 94 87 C2
                CE 80 DC 53 92 B6 FA B3
                BE 0F 71 BD 75 7D 68 B5
                D4 1F CB 37 6D 8F 08 7B
                3A 24 68 58 25 AF D0 F5
Verify signature using original message: "jumps over the lazy dog"
        Operation successful
Verify signature using modified message: "!umps over the lazy dog"
        Successfully detected modified message
Generating RSA encryption key
        Operation successful
Encrypting message: "Top secret" using RSA key: 257
        Operation successful
        Encrypted message:
                6A D6 90 8E F4 71 04 C6
                3D 70 E0 80 DB 32 FD 7D
                63 9B 9B B4 A2 98 F4 E2
                F7 CB 63 A9 AB 30 1D 25
                47 05 58 D4 D4 08 69 E8
                56 E6 F3 D8 EC A9 0B 6F
                07 E5 25 AA 7C D9 92 90
                3E 62 6A 7A 5D 6E E0 B8
                04 41 2A 28 3A C7 3A DA
                CF E6 DA 8D 41 C1 2A 69
                69 81 3C F3 63 11 9D 11
                D7 67 57 7F 23 A4 3B C2
                09 7B DE D4 53 72 C9 FC
                6E FB A4 D3 31 4E 89 5C
                AF B4 74 01 DC 2A 0F E9
                B8 F1 23 B4 11 04 D7 C0
Decrypting message using RSA key: 257
        Operation successful
        Decrypted message: "Top secret"
Exporting public key: 256
        Operation successful
        Public key bytes:
                04 9F 5B F2 4F 70 5A C6
                17 1E 9B F9 90 57 45 E1
                51 87 CB 07 A2 2E AB 6E
                7B 4F 31 01 CA C3 0D D0
                87 2B 2A 0C F7 CD 75 8D
                61 64 AF FD 96 6B B6 69
                6F 15 EC 10 73 BF 50 52
                23 2E E4 EC A5 19 F5 9B
                42
Destroying signing key: 256
        Operation successful
Destroying encryption key: 257
        Operation successful
#
```

10.4.3 Test Trusted Firmware-A

You can run the TF-A test on the Buildroot platform.

Procedure

1. After building Buildroot, run the following commands:

```
export TFTF_TESTS=true
./run_docker.sh build-tftf-tests.sh all with_reqs
```

2. Run Buildroot as normal. The test results display on **terminal_uart_ap**.

Results

The following example shows typical TF-A test results:


Note

The boot process does not fully complete when running the TF-A tests.

```
NOTICE:  Booting trusted firmware test framework
NOTICE:  Built : 17:01:31, Nov  6 2024
NOTICE:  v2.11(tc,release):v2.11.0-77-gf13410b

NOTICE:  Running at NS-EL2
NOTICE:  Starting a new test session
--
Running test suite 'Framework Validation'
Description: Validate the core features of the test framework

> Executing 'NVM support'
  TEST COMPLETE                                                 Passed

> Executing 'NVM serialisation'
  TEST COMPLETE                                                 Passed

> Executing 'Events API'
  TEST COMPLETE                                                 Passed

> Executing 'IRQ handling'
  TEST COMPLETE                                                 Passed

> Executing 'SGI support'
  TEST COMPLETE                                                 Passed

--
Running test suite 'Timer framework Validation'
Description: Validate the timer driver and timer framework

> Executing 'Verify the timer interrupt generation'
  TEST COMPLETE                                                 Passed

> Executing 'Target timer to a power down cpu'
  TEST COMPLETE                                                 Passed

> Executing 'Test scenario where multiple CPUs call same timeout'
  TEST COMPLETE                                                 Passed


--
.
.
.
> Executing 'Test Realm creation with LPA2 enabled but FEAT_LPA2 absent on platform'
  TEST COMPLETE                                                   Skipped
FEAT_RME not supported

****************************** Summary ******************************
> Test suite 'Framework Validation'
                                                                  Passed
> Test suite 'Timer framework Validation'
                                                                  Passed
> Test suite 'Boot requirement tests'
                                                                  Passed
```

```
> Test suite 'PSCI Version'                                                  Passed
> Test suite 'PSCI Affinity Info'                                            Passed
> Test suite 'CPU Hotplug'                                                   Passed
> Test suite 'PSCI CPU Suspend'                                              Passed
> Test suite 'PSCI STAT'                                                     Passed
> Test suite 'PSCI NODE_HW_STATE'                                            Passed
> Test suite 'PSCI Features'                                                 Passed
> Test suite 'PSCI MIGRATE_INFO_TYPE'                                        Passed
> Test suite 'PSCI mem_protect_check'                                        Passed
> Test suite 'SDEI'                                                          Passed
> Test suite 'Runtime Instrumentation Validation'                            Passed
> Test suite 'TRNG'                                                          Passed
> Test suite 'EM-ABI'                                                        Passed
> Test suite 'IRQ support in TSP'                                            Passed
> Test suite 'TSP handler standard functions result test'                    Passed
> Test suite 'Stress test TSP functionality'                                 Passed
> Test suite 'TSP PSTATE test'                                               Passed
> Test suite 'EL3 power state parser validation'                             Passed
> Test suite 'State switch'                                                  Passed
> Test suite 'CPU extensions'                                                Passed
> Test suite 'ARM_ARCH_SVC'                                                  Passed
> Test suite 'Performance tests'                                             Passed
> Test suite 'SMC calling convention'                                        Passed
> Test suite 'Query runtime services'                                        Passed
> Test suite 'FF-A Setup and Discovery'                                      Passed
> Test suite 'FF-A SMCCC compliance'                                         Passed
> Test suite 'FF-A Direct messaging'                                         Passed
> Test suite 'FF-A Group0 interrupts'                                        Passed
> Test suite 'FF-A Power management'                                         Passed
> Test suite 'FF-A Memory Sharing'                                           Passed
> Test suite 'SIMD context switch tests'                                     Passed
> Test suite 'FF-A Notifications'                                            Passed
> Test suite 'FF-A Indirect Messaging'                                       Passed
> Test suite 'PMU Leakage'                                                   Passed
> Test suite 'DebugFS'                                                       Passed
> Test suite 'RMI and SPM tests'
```

```
                                                                    Passed
> Test suite 'Realm payload at EL1'
                                                                    Passed
================================
Tests Skipped : 173
Tests Passed  : 99
Tests Failed  : 0
Tests Crashed : 0
Total tests   : 272
================================
NOTICE:  Exiting tests.
```

10.4.4 Run the kernel self test

This procedure describes how to run the kernel self test.

Procedure

1. In the **terminal_uart_ap** window, navigate to the `/usr/bin/selftest` directory.
2. To run all the tests at once, enter the following command:

```
./run_kselftest.sh --summary
```


Note

- Running all the kernel tests takes a considerable amount of time. Alternatively, run the tests individually. For information about how to run the tests individually, run the command `./run_kselftest.sh --help`.
- Because the *Kernel Same page Merging* (KSM) driver is not a part of the Lumex kernel, one of the MTE Kselftests fails for the `check_ksm_options` test.

Results

The following example shows typical test output:

```
# ./run_kselftest.sh --summary
[  520.082187][  T177] kselftest: Running tests in arm64
TAP version 13
1..17
# selftests: arm64: check_prctl
ok 1 selftests: arm64: check_prctl
# selftests: arm64: check_gcr_el1_cswitch
ok 2 selftests: arm64: check_gcr_el1_cswitch
# selftests: arm64: check_ksm_options
not ok 3 selftests: arm64: check_ksm_options # exit=1
# selftests: arm64: check_tags_inclusion
ok 4 selftests: arm64: check_tags_inclusion
# selftests: arm64: check_user_mem
ok 5 selftests: arm64: check_user_mem
# selftests: arm64: check_mmap_options
ok 6 selftests: arm64: check_mmap_options
# selftests: arm64: check_child_memory
ok 7 selftests: arm64: check_child_memory
# selftests: arm64: check_buffer_fill
ok 8 selftests: arm64: check_buffer_fill
# selftests: arm64: btitest
ok 9 selftests: arm64: btitest
# selftests: arm64: nobtitest
ok 10 selftests: arm64: nobtitest
```

```
# selftests: arm64: pac
ok 11 selftests: arm64: pac
# selftests: arm64: fp-stress
ok 12 selftests: arm64: fp-stress
# selftests: arm64: sve-ptrace
ok 13 selftests: arm64: sve-ptrace
# selftests: arm64: sve-probe-vls
ok 14 selftests: arm64: sve-probe-vls
# selftests: arm64: vec-syscfg
ok 15 selftests: arm64: vec-syscfg
# selftests: arm64: za-fork
ok 16 selftests: arm64: za-fork
# selftests: arm64: za-ptrace
ok 17 selftests: arm64: za-ptrace
#
```

10.4.5 Test MPAM

This procedure describes how to run the *Memory System Resource Partitioning and Monitoring* (MPAM) confidence test.

Procedure

1. Verify that the hardware and the software requirements for the MPAM feature by running the following command on the **terminal_uart_ap** terminal.

```
testing_mpam.sh tc4 fvp
```

Because this script is in the `/bin` folder and part of the default `$PATH` environment variable, you can run this command from any directory.

Results

The following example shows typical test output for the FVP:

```
# testing_mpam.sh tc4 fvp
Testing the number of partitions supported.  It should be 0-63
Pass
Partition 0 is the default partition to which all tasks will be assigned.  Checking
 if task 1 is assigned to partition 0
Pass
Checking DSU directory exists
Pass
Testing the number of bits required to set the cache portion bitmap. It should be 8
Pass
Testing the default cpbm configured in the DSU for all the partitions.  It should be
 0-7 for all the partitions
[ 1504.362904][  T198] MPAM_arch: mpam_msc_sel_partid(): PART_SEL: 0x0
Pass
Setting the cpbm 4-5 in DSU for partition 6 and reading it back
[ 1504.363102][  T187] MPAM_arch: mpam_msc_sel_partid(): PART_SEL: 0x6
[ 1504.363111][  T187] MPAM_arch: mpam_msc_set_cpbm(): CPBM: 0x30 @ffff800080403000
[ 1504.363905][  T199] MPAM_arch: mpam_msc_sel_partid(): PART_SEL: 0x6
Pass
#
```

10.4.6 Test MPMM on FVP

Maximum Power Mitigation Mechanism (MPMM) is a power management feature that detects and limits high activity events, specifically high-power load/store events and vector unit instructions. This procedure describes how to run the MPMM confidence test on the FVP.

About this task

You can leverage the functionality of the MPMM module in the SCP firmware to do the following:

- Set the proper gear for each core based on the workload. You can verify this functionality by checking the `INFO` level SCP logs while executing the `vector_workload` test application on the **terminal_uart_ap** window as follows:

  ```
  vector_workload
  ```

- Enforce the maximum clock frequency for a group of cores of the same type, based on the current gear set for each core in that group. You can exercise this functionality by running the provided shell script `test_mpmm.sh` which runs `vector_workload` on the different cores. This test ensures that the maximum clock frequency for a group of cores of the same type does not exceed the values set in *Perf Constraint Lookup Table* (PCT) of the MPMM module in the SCP firmware.

Procedure

Enter the following command in the **terminal_uart_ap** window:

```
test_mpmm.sh tc4 fvp
```

Results

The following example shows typical test output:

```
# test_mpmm.sh tc4 fvp

Testing MPMM in FVP

Testing the MPMM of Arm C1-Nano cores
*****************************
According to the PCT, the max frequency should be 2152000
Current set frequency of the cpu0 is 768000
PASS

Starting a vector intensive workload on cpu0
According to the PCT, the max frequency should be 2152000
Current set frequency of the cpu0 is 2152000
PASS

Starting a vector intensive workload on cpu1
According to the PCT, the max frequency should be 1844000
Current set frequency of the cpu0 is 1844000
PASS

Testing the MPMM of Arm C1-Pro cores
*****************************
According to the PCT, the max frequency should be 2650000
Current set frequency of the cpu2 is 1893000
PASS

Starting a vector intensive workload on cpu2
```

```
According to the PCT, the max frequency should be 2271000
Current set frequency of the cpu2 is 2271000
PASS

Starting a vector intensive workload on cpu3
According to the PCT, the max frequency should be 1893000
Current set frequency of the cpu2 is 1893000
PASS

Starting a vector intensive workload on cpu4
According to the PCT, the max frequency should be 1419000
Current set frequency of the cpu2 is 1419000
PASS

Starting a vector intensive workload on cpu5
According to the PCT, the max frequency should be 1419000
Current set frequency of the cpu2 is 1419000
PASS

Testing the MPMM of Arm C1-Ultra cores
*****************************
According to the PCT, the max frequency should be 3047000
Current set frequency of the cpu6 is 2176000
PASS

Starting a vector intensive workload on cpu6
According to the PCT, the max frequency should be 2612000
Current set frequency of the cpu6 is 2176000
PASS

Starting a vector intensive workload on cpu7
According to the PCT, the max frequency should be 2176000
Current set frequency of the cpu6 is 2176000
PASS
#
```

10.4.7 Test the rotational scheduler

Rotational scheduling is enabled by default and ensures that all the tasks finish at approximately at the same time with a minimum of idle time on any CPU. This procedure describes how to run the rotational scheduler confidence test.

Procedure

1. Log into the **terminal_uart_ap** with username `root`.
2. Run the following command:

```
test_rotational_scheduler.sh
```

Results

The following example shows typical test output:

```
# test_rotational_scheduler.sh
    Enable The Rotational Scheduler
    Pass

    Checking the value of max_latency_us
    Pass

    Checking the value of max_residency_us
    Pass
```

```
    Checking the value of min_residency_us
    Pass

    Checking the value of hysteresis_active_tick
    Pass

    #
```

10.4.8 Test SCMI

This procedure describes how to run the *System Control and Management Interface* (SCMI) confidence test.

Procedure

1. Build the Buildroot distribution, and when you prepare the compilation environment as described in the procedure, add the following command before you run `./setup.sh`:

   ```
   export SCMI_TESTS=true
   ```

2. After building the Buildroot distribution and running the FVP, enter the following command on the **terminal_uart_ap**:

   ```
   ./scmi_test_agent
   ```

   The test generates a log file with the filename `arm_scmi_test_log.txt`.

Results

The following example shows typical test output:


Note

The test failures on test cases 409, 413, and 517 are expected and due to known issues.

```
SCMI unit tests
---------------

::

    # cat arm_scmi_test_log.txt
           **** SCMI Compliance Suite ****
      Using SCMI kernel Raw transport rooted at:/sys/kernel/debug/scmi/0/raw
      Resetting SCMI kernel Raw queues.
      Using *strict* SCMI protocol version checking

              *** Starting BASE tests ***
    101: Base protocol version check
        [Check 1] Query protocol version
          MSG HDR        : 0x00004000
          NUM PARAM      : 0
          CHECK STATUS   : PASSED [SCMI_STATUS_SUCCES]
          CHECK HEADER   : PASSED [0x00004000]
          RETURN COUNT   : 1
          RETURN[00]     : 0x00020000
          VERSION        : 0x00020000                              : CONFORMANT
    102: Base protocol attributes check
        [Check 1] Query protocol attributes
```

```
        MSG HDR         : 0x00044001
        NUM PARAM       : 0
        CHECK STATUS    : PASSED [SCMI_STATUS_SUCCES]
        CHECK HEADER    : PASSED [0x00044001]
        RETURN COUNT    : 1
        RETURN[00]      : 0x00000204
        CHECK RSVD BITS: PASSED
        CHECK NUM AGENTS: PASSED [0x00000002]
        CHECK NUM PROTOCOLS: PASSED [0x00000004]                : CONFORMANT
        .
        .
        .
  517: Clock config set check
        NUM CLOCKS      : 3
      CLOCK ID: 0
      [Check 1] Set config with attributes :0
        MSG HDR         : 0x05885007
        NUM PARAM       : 2
        PARAMETER[00]   : 0x00000000
        PARAMETER[01]   : 0x00000000
        CHECK HEADER    : PASSED [0x05885007]
        CHECK STATUS    : FAILED
            EXPECTED    : SCMI_DENIED_ERROR
            RECEIVED    : SCMI_NOT_SUPPORTED
        CHECK STATUS    : PASSED [SCMI_NOT_SUPPORTED]
      CLOCK ID: 1
      [Check 1] Set config with attributes :1
        MSG HDR         : 0x058c5007
        NUM PARAM       : 2
        PARAMETER[00]   : 0x00000001
        PARAMETER[01]   : 0x00000001
        CHECK HEADER    : PASSED [0x058c5007]
        CHECK STATUS    : FAILED
            EXPECTED    : SCMI_DENIED_ERROR
            RECEIVED    : SCMI_STATUS_SUCCES
        CHECK STATUS    : FAILED
            EXPECTED    : SCMI_NOT_SUPPORTED
            RECEIVED    : SCMI_STATUS_SUCCES
        CHECK STATUS    : PASSED [SCMI_STATUS_SUCCES]
        RETURN COUNT    : 0
      [Check 2] Verify the changed attribute
        MSG HDR         : 0x05905003
        NUM PARAM       : 1
        PARAMETER[00]   : 0x00000001
        CHECK STATUS    : PASSED [SCMI_STATUS_SUCCES]
        CHECK HEADER    : PASSED [0x05905003]
        CHECK RSVD BITS: PASSED
        RETURN COUNT    : 5
        RETURN[00]      : 0x00000001
        RETURN[01]      : 0x45584950
        RETURN[02]      : 0x00305f4c
        RETURN[03]      : 0x00000000
        RETURN[04]      : 0x00000000
        CHECK CLOCK STATUS : PASSED [0x00000001]
      [Check 3] Restore the original attributes :0
        MSG HDR         : 0x05945007
        NUM PARAM       : 2
        PARAMETER[00]   : 0x00000001
        PARAMETER[01]   : 0x00000000
        CHECK HEADER    : PASSED [0x05945007]
        CHECK STATUS    : FAILED
            EXPECTED    : SCMI_STATUS_SUCCES
            RECEIVED    : SCMI_NOT_SUPPORTED                : NON CONFORMANT

          *** Starting SENSOR tests ***
  Calling agent have no access to SENSOR protocol

          *** Starting RESET tests ***
  Calling agent have no access to RESET protocol

          *** Starting VOLTAGE tests ***
```

```
 Calling agent have no access to Voltage protocol
**************************************************
  TOTAL TESTS: 86    PASSED: 75    FAILED: 1    SKIPPED: 10
**************************************************

        **** SCMI tests complete ****
```

10.4.9 Test processor core capabilities

This procedure describes how to run the confidence tests for processor core capabilities.

Procedure

The Buildroot build variant provides a script that validates the advertizement for the `FEAT_AFP`, `FEAT_ECV`, and `FEAT_WFxT` processor core hardware capabilities.
On the **terminal_uart_ap** terminal, enter the following command:

```
test_feats_arch.sh
```

Results

The following example shows typical test output:

```
# test_feats_arch.sh
Testing FEAT_AFP HW CAP
Pass

Testing FEAT_ECV HW CAP
Pass

Testing FEAT_WFXT HW CAP
Pass

#
```

11. Troubleshooting

Arm provides potential solutions to the most common problems experienced by developers related to the host development environment.

This list is not exhaustive. Other resources, for example Arm Developer, contain additional solutions not covered in this document.

11.1 Cannot connect to Docker

The Docker service must be running for Lumex to work correctly.

Docker not started

If the Docker instance has not been started or has been stopped, the error message `Cannot Connect to a Docker Daemon` displays when Lumex tries to connect to it.

Solution

Ensure the Docker service is running, that the permissions are correct, and that the user group membership is properly configured. For more information, see Set up the build environment.

11.2 Docker connection refused

The Docker service must be running and available for the system to function properly.

Docker in an inconsistent state

If Lumex cannot connect to Docker, the error message `transport: dial unix /var/run/docker/containerd/docker-containerd.sock: connect: connection refused` displays.

Solution

Restart the Docker service by entering the command `sudo systemctl restart docker`.

Proprietary Notice

This document is protected by copyright and other related rights and the use 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 Limited ("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 the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. 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, any patents, copyrights, trade secrets, trademarks, or other rights.

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.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted 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.

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 this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0000-04	24 September 2025	Non-Confidential	Public release for Lumex
0000-03	5 August 2024	Confidential	Third release for Lumex
0000-02	1 December 2024	Confidential	Second release for Lumex, Support for Android 15, Debian, BCC/DICE, and rotational scheduler
0000-01	9 May 2024	Confidential	First release for Lumex

Change history

For information about the functional changes to Arm® Lumex Reference Software, see the Arm® Lumex Reference Software Release Notes.

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.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. 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

We recommend the following. If you do not follow these recommendations your system might not work.

Warning

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or harming yourself.

This information is important and needs your attention.

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

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

Useful resources

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

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm Statistical Profiling Extension: Performance Analysis Methodology White Paper	109429	Non-Confidential
Arm® C1-DynamIQ™ Shared Unit Technical Reference Manual	107804	Non-Confidential
Arm® Client Platform Security Architecture Firmware Framework, version 1.0	DEN 0063	Non-Confidential
Arm® Development Studio User Guide	101470	Non-Confidential
Arm® MMU L1 System Memory Management Unit Technical Reference Manual	107957	Non-Confidential
Fast Models Fixed Virtual Platforms (FVP) Reference Guide	100966	Non-Confidential
Fast Models Reference Guide	100964	Non-Confidential
What is the Difference between Arm Statistical Profiling Extension and Performance Monitor Unit Events?	KA005848	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential
Arm® Corstone™ Reference Systems Architecture Specification Ma1	102803	Non-Confidential
Arm® Debug Interface Architecture Specification ADIv6.0	IHI 0074	Non-Confidential

Non-Arm resources	Document ID	Organization
Android Common Kernel (ACK)	–	Google
Android NDK downloads page	–	Google
Android Simpleperf tool README	–	Google
Android Simpleperf tool command reference	–	Google
Android Simpleperf tool documentation	–	Google
Android Studio Terms and conditions	–	Google
Android documentation	–	Google
AutoFDO GitHub repository	–	AutoFDO GitHub repository
AutoFDO tool	–	Google
Buildroot GitHub	–	Buildroot
Debian GitHub	–	Debian
Hafnium	–	TrustedFirmware
OP-TEE	–	OP-TEE Project
Perfetto developer docs	–	Perfetto developer docs
Perfetto web UI	–	Perfetto web UI

Non-Arm resources	Document ID	Organization
RSE Firmware	–	RSE Firmware
Support for Arm Statistical Profiling Extension within Perf tools	–	Linux®
TFLite Model Benchmark Tool README	–	TFLite Model Benchmark Tool README
TensorFlow	–	TensorFlow
Trusted Firmware-A documentation	–	TrustedFirmware
Trusted Firmware-M documentation	–	TrustedFirmware
Trusted Services	–	TrustedFirmware
Trusty	–	Android Open Source Project
U-boot	–	DENX Software Engineering
perf documentation	–	Bootlin
perf wiki	–	Linux Kernel Organization