
corstone1000

Arm Limited

Dec 23, 2024

CONTENTS

1 Disclaimer

1

DISCLAIMER

Arm reference solutions are Arm public example software projects that track and pull upstream components, incorporating their respective security fixes published over time. Arm partners are responsible for ensuring that the components they use contain all the required security fixes, if and when they deploy a product derived from Arm reference solutions.

1.1 Software Architecture

1.1.1 Arm Corstone-1000

Arm Corstone-1000 is a reference solution for IoT devices. It is part of Total Solution for IoT which consists of hardware and software reference implementation.

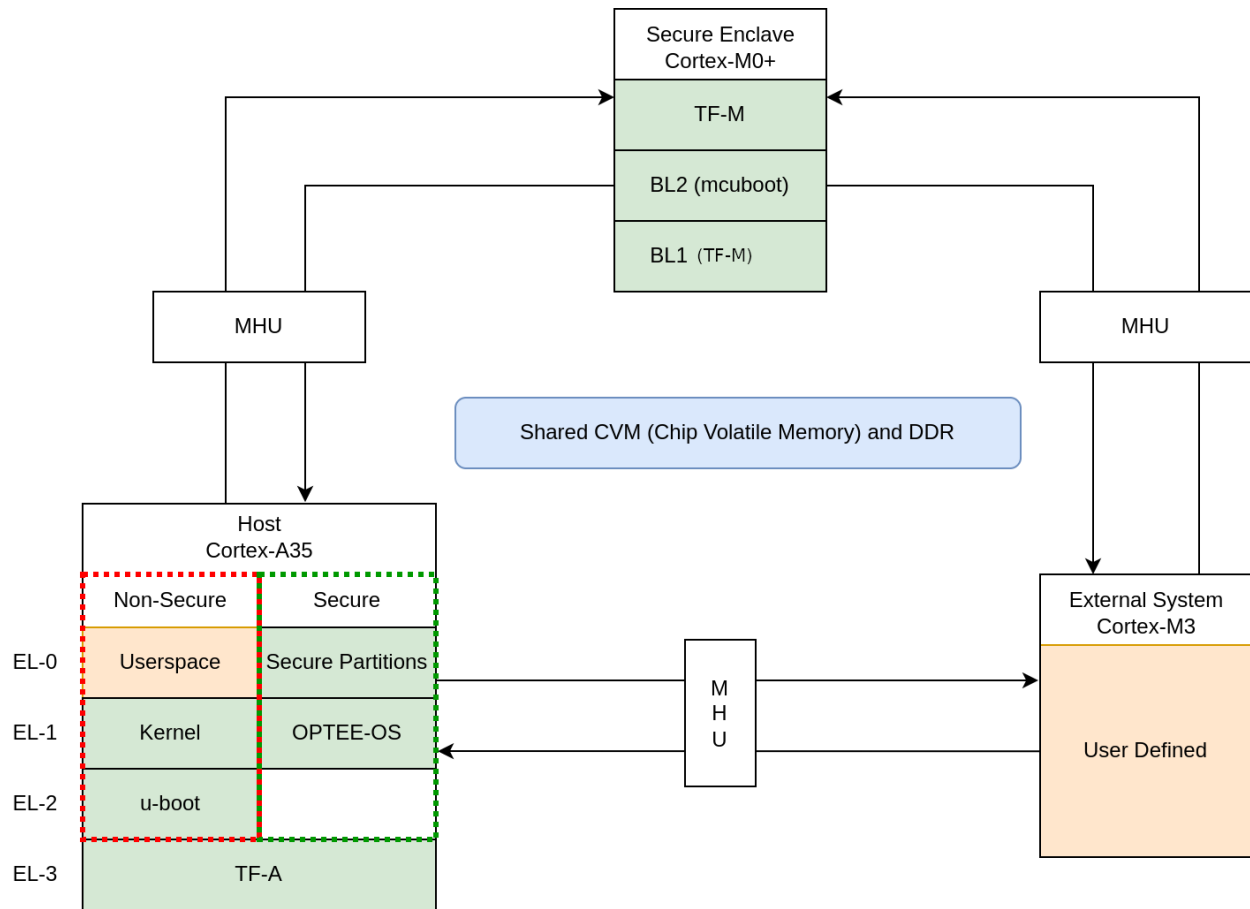
Corstone-1000 software plus hardware reference solution is PSA Level-2 ready certified ([PSA L2 Ready](#)) as well as System Ready IR certified ([SRIR cert](#)). More information on the Corstone-1000 subsystem product and design can be found at: [Arm Corstone-1000 Software](#) and [Arm Corstone-1000 Technical Overview](#).

This readme explicitly focuses on the software part of the solution and provides internal details on the software components. The reference software package of the platform can be retrieved following instructions present in the user-guide document.

1.1.2 Design Overview

The software architecture of Corstone-1000 platform is a reference implementation of Platform Security Architecture (PSA) which provides framework to build secure IoT devices.

The base system architecture of the platform is created from three different types of systems: Secure Enclave, Host and External System. Each subsystem provides different functionality to overall SoC.



The Secure Enclave System, provides PSA Root of Trust (RoT) and cryptographic functions. It is based on an Cortex-M0+ processor, CC312 Cryptographic Accelerator and peripherals, such as watchdog and secure flash. Software running on the Secure Enclave is isolated via hardware for enhanced security. Communication with the Secure Enclave is achieved using Message Handling Units (MHUs) and shared memory. On system power on, the Secure Enclave boots first. Its software comprises of a ROM code (TF-M BL1), MCUBoot BL2, and TrustedFirmware-M (TF-M) as runtime software. The software design on Secure Enclave follows Firmware Framework for M class processor (FF-M) specification.

The Host System is based on ARM Cortex-A35 processor with standardized peripherals to allow for the booting of a Linux OS. The Cortex-A35 has the TrustZone technology that allows secure and non-secure security states in the processor. The software design in the Host System follows Firmware Framework for A class processor (FF-A) specification. The boot process follows Trusted Boot Base Requirement (TBBR). The Host Subsystem is taken out of reset by the Secure Enclave system during its final stages of the initialization. The Host subsystem runs FF-A Secure Partitions (based on [Trusted Services](#)) and OPTEE-OS (OPTEE-OS) in the secure world, and U-Boot (U-Boot repo) and linux (linux repo) in the non-secure world. The communication between non-secure and the secure world is performed via FF-A messages.

An external system is intended to implement use-case specific functionality. The system is based on Cortex-M3 and run RTX RTOS. Communication between the external system and Host (Cortex-A35) can be performed using MHU

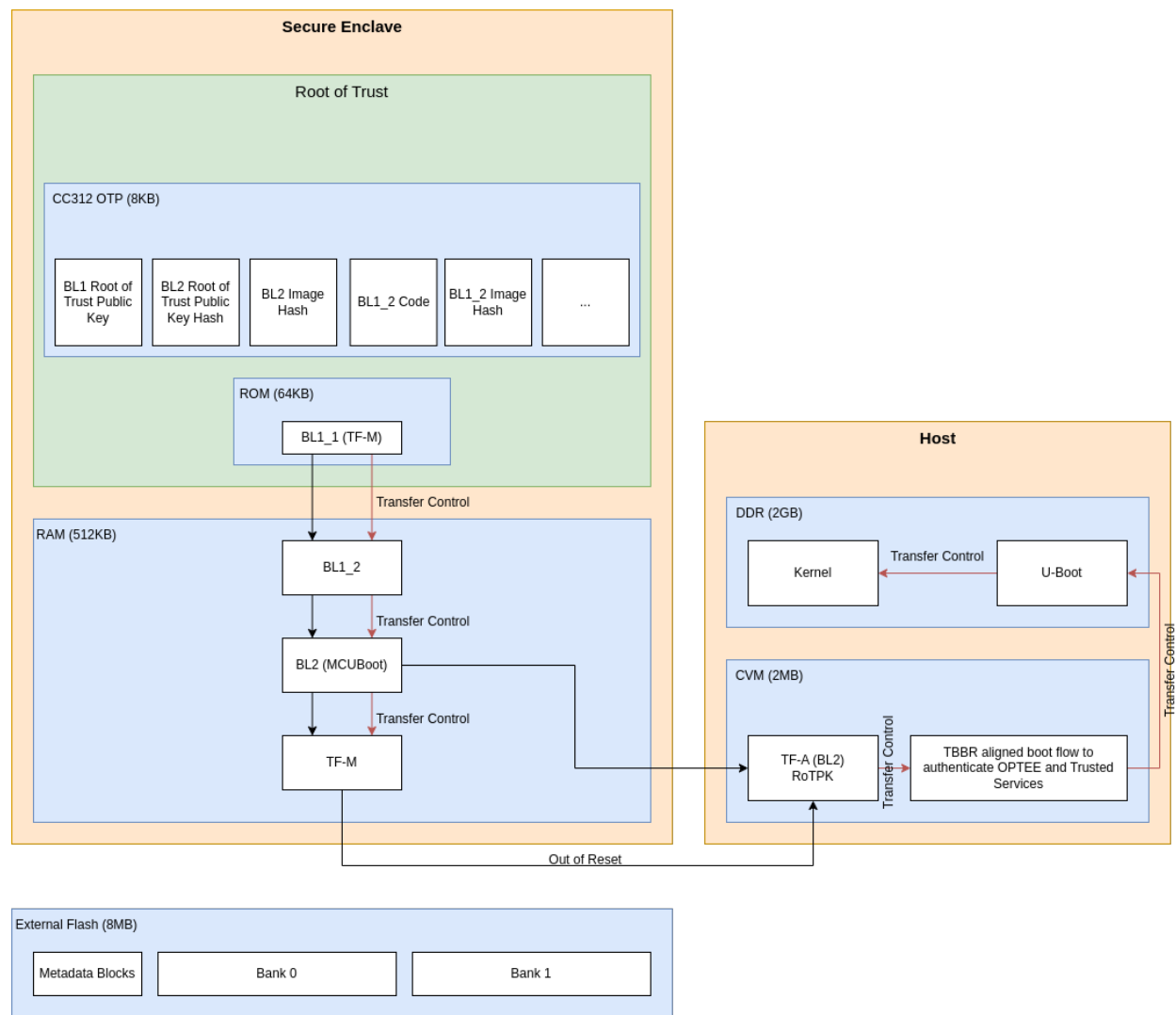
as transport mechanism. The current software release supports switching on and off the external system. Support for OpenAMP-based communication is under development.

Overall, the Corstone-1000 architecture is designed to cover a range of Power, Performance, and Area (PPA) applications, and enable extension for use-case specific applications, for example, sensors, cloud connectivity, and edge computing.

1.1.3 Secure Boot Chain

For the security of a device, it is essential that only authorized software should run on the device. The Corstone-1000 boot uses a Secure Boot Chain process where an already authenticated image verifies and loads the following software in the chain. For the boot chain process to work, the start of the chain should be trusted, forming the Root of Trust (RoT) of the device. The RoT of the device is immutable in nature and encoded into the device by the device owner before it is deployed into the field. In Corstone-1000, the content of the ROM and CC312 OTP (One Time Programmable) memory forms the RoT.

Verification of an image can happen either by comparing the computed and stored hashes, or by checking the signature of the image if the image is signed.



It is a lengthy chain to boot the software on Corstone-1000. On power on, the Secure Enclave starts executing BL1_1

code from the ROM which is the RoT of the device. The BL1_1 is the immutable bootloader of the system, it handles the provisioning on the first boot, hardware initialization and verification of the next stage.

The BL1_2 code, hashes and keys are written into the OTP during the provisioning. The next bootstage is the BL1_2 which is copied from the OTP into the RAM. The BL1_1 also compares the BL1_2 hash with the hash saved to the OTP. The BL1_2 verifies and transfers control to the next bootstage which is the BL2. During the verification, the BL1_2 compares the BL2 image's computed hash with the BL2 hash in the OTP. The BL2 is MCUBoot in the system. BL2 can provision additional keys on the first boot and it authenticates the initial bootloader of the host (Host TF-A BL2) and TF-M by checking the signatures of the images. The MCUBoot handles the image verification the following way:

- Load image from a non-volatile memory to dynamic RAM.
- The public key present in the image header is validated by comparing with the hash. Depending on the image, the hash of the public key is either stored in the OTP or part of the software which is being already verified in the previous stages.
- The image is validated using the public key.

The execution control is passed to TF-M after the verification. TF-M being the runtime executable of the Secure Enclave which initializes itself and, at the end, brings the host CPU out of rest.

The TF-M BL1 design details and reasoning can be found in the [TF-M design documents](#).

The Corstone-1000 has some differences compared to this design due to memory (OTP/ROM) limitations:

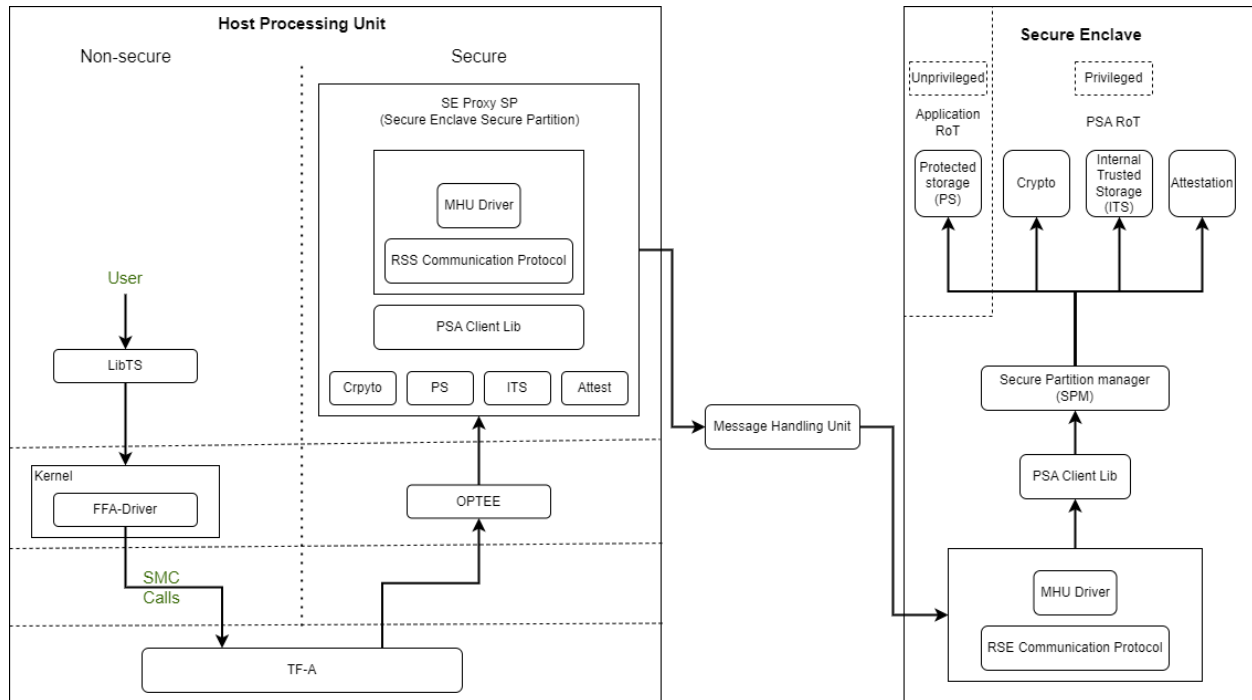
- The provisioning bundle that contains the BL1_2 code is located in the ROM. This means the BL1_2 cannot be updated during provisioning time.
- The BL1_1 handles most of the hardware initialization instead of the BL1_2. This results in a bigger BL1_1 code size than needed.
- The BL1_2 does not use the post-quantum LMS verification. The BL2 is verified by comparing the computed hash to the hash which is stored in the OTP. This means the BL2 is not updatable.

The host follows the boot standard defined in the [TBRR](#) to authenticate the secure and non-secure software.

For UEFI Secure Boot, authenticated variables can be accessed from the secure flash. The feature has been integrated in U-Boot, which authenticates the images as per the UEFI specification before executing them.

1.1.4 Secure Services

Corstone-1000 is unique in providing a secure environment to run a secure workload. The platform has TrustZone technology in the Host subsystem but it also has hardware isolated Secure Enclave environment to run such secure workloads. In Corstone-1000, known Secure Services such as Crypto, Protected Storage, Internal Trusted Storage and Attestation are available via PSA Functional APIs in TF-M. There is no difference for a user communicating to these services which are running on a Secure Enclave instead of the secure world of the host subsystem. The below diagram presents the data flow path for such calls.



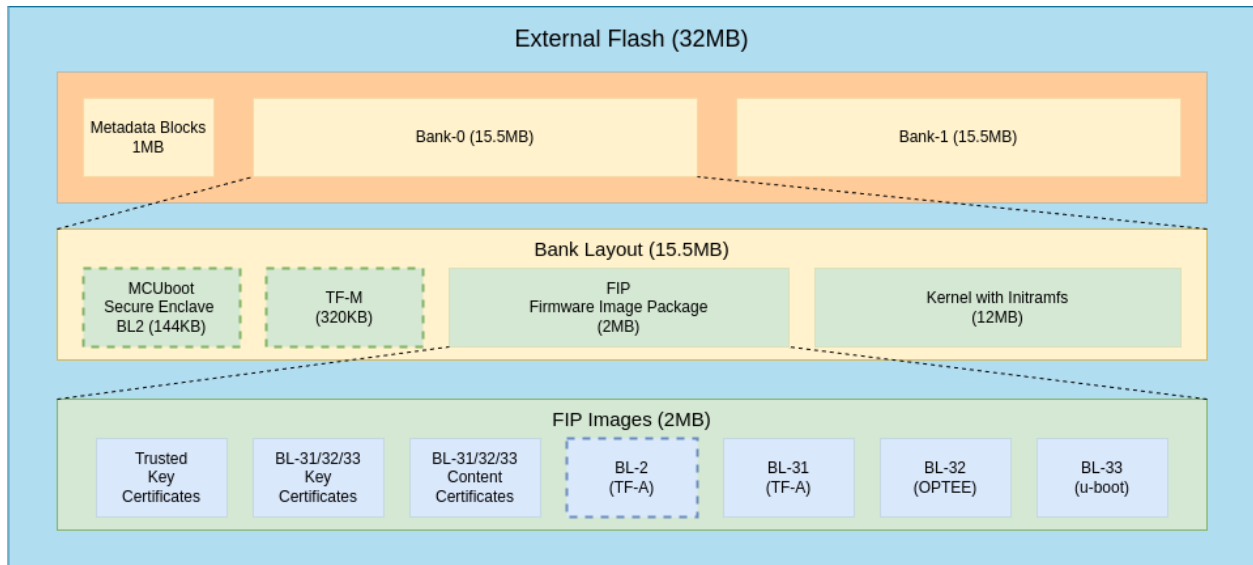
The SE Proxy SP (Secure Enclave Proxy Secure Partition) is a proxy partition managed by OPTEE which forwards such calls to the Secure Enclave. The solution relies on the [RSE communication protocol](#) which is a lightweight serialization of the `psa_call()` API. It can use shared memory and MHU interrupts as a doorbell for communication between two cores but currently the whole message is forwarded through the MHU channels in Corstone-1000. Corstone-1000 implements isolation level 2. Cortex-M0+ MPU (Memory Protection Unit) is used to implement isolation level 2.

For a user to define its own secure service, both the options of the host secure world or secure enclave are available. It's a trade-off between lower latency vs higher security. Services running on a Secure Enclave are secure by real hardware isolation but have a higher latency path. In the second scenario, the services running on the secure world of the host subsystem have lower latency but virtual hardware isolation created by TrustZone technology.

1.1.5 Secure Firmware Update

Apart from always booting the authorized images, it is also essential that the device only accepts the authorized (signed) images in the firmware update process. Corstone-1000 supports OTA (Over the Air) firmware updates and follows Platform Security Firmware Update specification (FWU).

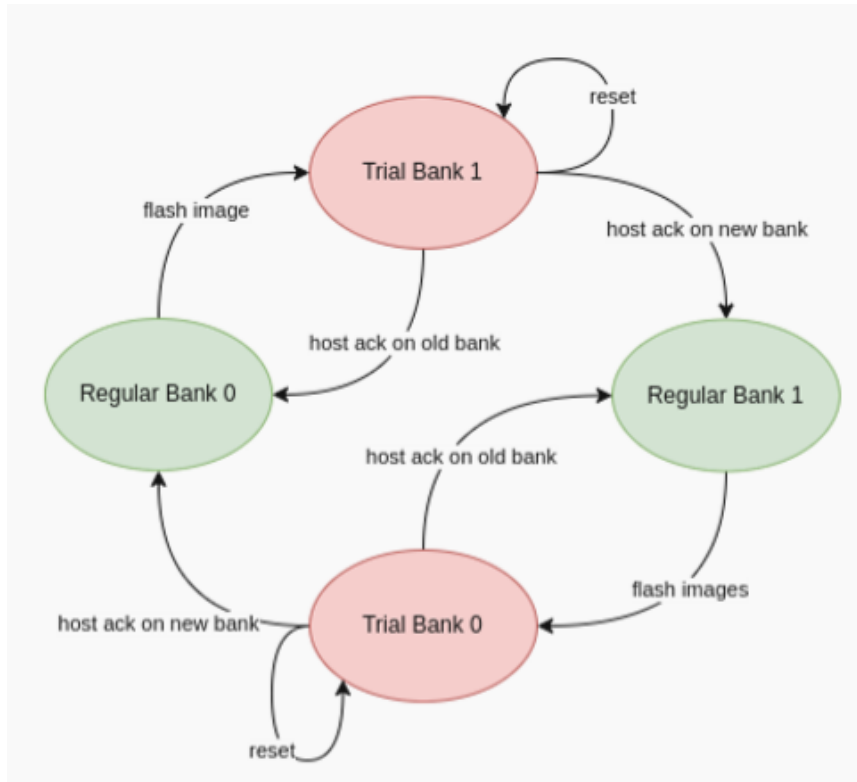
As standardized into [FWU](#), the external flash is divided into two banks of which one bank has currently running images and the other bank is used for staging new images. There are four updatable units, i.e. Secure Enclave's BL2 and TF-M, and Host's FIP (Firmware Image Package) and Kernel Image (the `initramfs` bundle). The new images are accepted in the form of a UEFI capsule.



When Firmware update is triggered, U-Boot verifies the capsule by checking the capsule signature, version number and size. Then it signals the Secure Enclave that can start writing UEFI capsule into the flash.

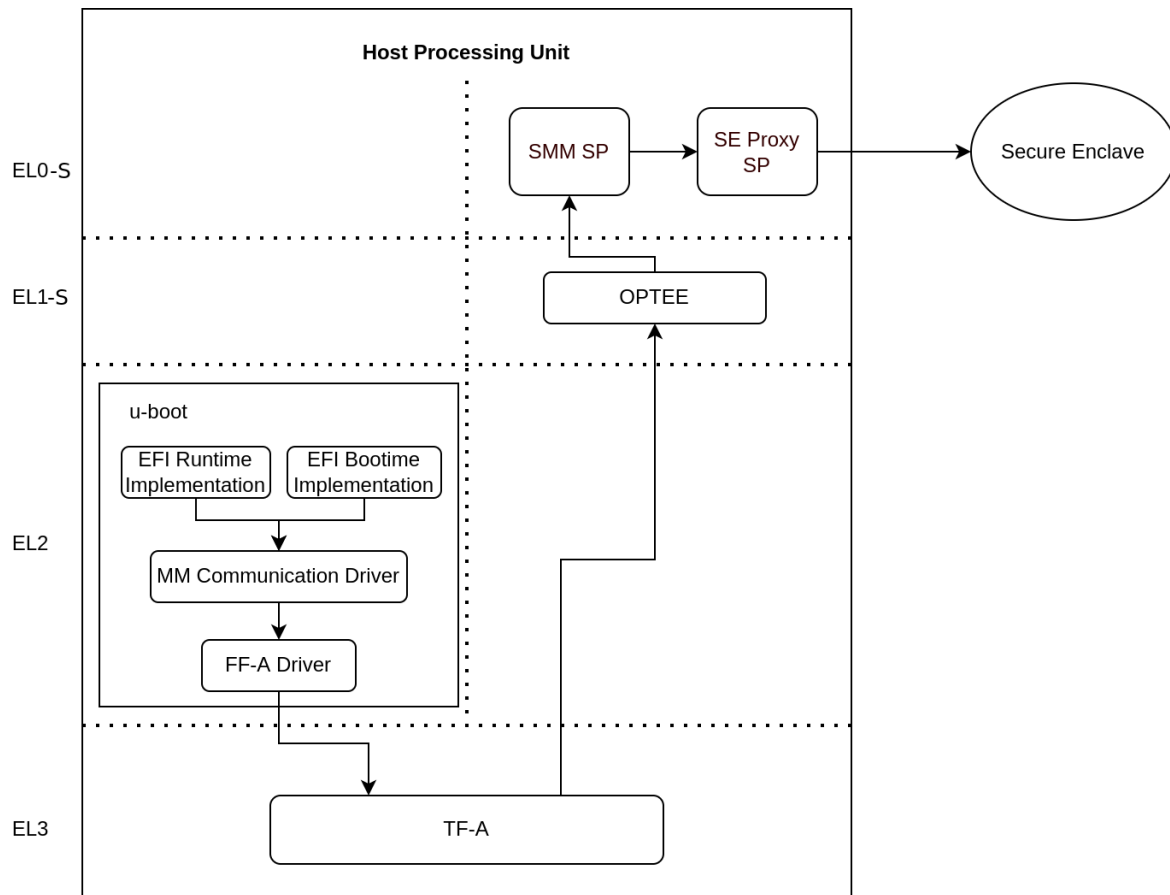
Once this operation finishes, Secure Enclave resets the entire system. The Metadata Block in the flash has the below firmware update state machine. TF-M runs an OTA service that is responsible for accepting and updating the images in the flash. The communication between the UEFI Capsule update subsystem and the OTA service follows the same data path explained above. The OTA service writes the new images to the passive bank after successful capsule verification. It changes the state of the system to trial state and triggers the reset.

Boot loaders in Secure Enclave and Host read the Metadata block to get the information on the boot bank. In the successful trial stage, the acknowledgment from the host moves the state of the system from trial to regular. Any failure in the trial stage or system hangs leads to a system reset. This is made sure by the use of watchdog hardware. The Secure Enclave's BL1 has the logic to identify multiple resets and eventually switch back to the previous good bank. The ability to revert to the previous bank is crucial to guarantee the availability of the device.



1.1.6 UEFI Runtime Support in U-Boot

Implementation of UEFI boottime and runtime APIs require variable storage. In Corstone-1000, these UEFI variables are stored in the Protected Storage service. The below diagram presents the data flow to store UEFI variables. The U-Boot implementation of the UEFI subsystem uses the U-Boot FF-A driver to communicate with the SMM Service in the secure world. The backend of the SMM service uses the proxy PS from the SE Proxy SP. From there on, the PS calls are forwarded to the Secure Enclave as explained above.



1.1.7 References

ARM Corstone-1000 Search

Arm security features

Copyright (c) 2022-2024, Arm Limited. All rights reserved.

1.2 Build, Flash and Run

1.2.1 Notice

The Corstone-1000 software stack uses the [Yocto Project](#) to build a tiny Linux distribution suitable for the Corstone-1000 platform (kernel and iniramfs filesystem less than 5 MB on the flash). The Yocto Project relies on the [BitBake](#) tool as its build tool. Please see [Yocto Project documentation](#) for more information.

1.2.2 Prerequisites

This guide assumes that your host machine is running Ubuntu 20.04 LTS (with sudo rights), with at least 32GB of free disk space and 16GB of RAM as minimum requirement.

The following prerequisites must be available on the host system:

- Git 1.8.3.1 or greater.
- Python 3.8.0 or greater.
- GNU Tar 1.28 or greater.
- GNU Compiler Collection 8.0 or greater.
- GNU Make 4.0 or greater.
- tmux.

Please follow the steps described in the Yocto mega manual:

- [Compatible Linux Distribution](#)
- [Build Host Packages](#)

1.2.3 Targets

The Corstone-1000 software stack can be run on:

- [Arm Corstone-1000 Ecosystem FVP \(Fixed Virtual Platform\)](#)
- [Arm Corstone-1000 for MPS3](#)

Important: Arm Corstone-1000 for MPS3 requires an additional 32 MB QSPI flash PMOD module. For more information see the [Application Note AN550 document](#).

1.2.4 Yocto Stable Branch

Corstone-1000 software stack is built on top of Yocto styhead release.

1.2.5 Software Components

Within the Yocto Project, each component included in the Corstone-1000 software stack is specified as a [BitBake recipe](#). The recipes specific to the Corstone-1000 BSP are located at: `$WORKSPACE/meta-arm/meta-arm-bsp/`.

Important: `$WORKSPACE` refers to the absolute path to your workspace where the *meta-arm* repository will be cloned. `$TARGET` is either `mps3` or `fvp`.

The Yocto machine config files for the Corstone-1000 FVP and MPS3 targets are:

- `$WORKSPACE/meta-arm/meta-arm-bsp/conf/machine/include/corstone1000.inc`
- `$WORKSPACE/meta-arm/meta-arm-bsp/conf/machine/corstone1000-$TARGET.conf`

Note: All the paths stated in this document are absolute paths.

Host Processor Components

Trusted Firmware-A

bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-bsp/trusted-firmware-a/trusted-firmware-a_%.bbappend
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-bsp/trusted-firmware-a/trusted-firmware-a_2.11.0.bb

Trusted Services

bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/trusted-services/libts_%.bbappend
bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/trusted-services/ts-psa-crypto-api-test_%.bbappend
bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/trusted-services/ts-psa-iat-api-test_%.bbappend
bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/trusted-services/ts-psa-its-api-test_%.bbappend
bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/trusted-services/ts-psa-ps-api-test_%.bbappend
bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/trusted-services/ts-sp-se-proxy_%.bbappend
bbappend	\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/trusted-services/ts-sp-smm-gateway_%.bbappend
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-security/trusted-services/libts_git.bb
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-security/trusted-services/ts-psa-crypto-api-test_git.bb
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-security/trusted-services/ts-psa-iat-api-test_git.bb
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-security/trusted-services/ts-psa-its-api-test_git.bb
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-security/trusted-services/ts-psa-ps-api-test_git.bb
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-security/trusted-services/ts-sp-smm-gateway.bb
Recipe	\$WORKSPACE/meta-arm/meta-arm/recipes-security/trusted-services/ts-sp-se-proxy.bb

OP-TEE

bbappend	<code>\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-security/optee/optee-os_4%.bbappend</code>
Recipe	<code>\$WORKSPACE/meta-arm/meta-arm/recipes-security/optee/optee-os_4.2.0.bb</code>

U-Boot

bbappend	<code>\$WORKSPACE/meta-arm/meta-arm/recipes-bsp/u-boot/u-boot%.bbappend</code>
bbappend	<code>\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-bsp/u-boot/u-boot%.bbappend</code>
Recipe	<code>\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-bsp/u-boot/u-boot_2023.07.02.bb</code>

Linux

The distribution is based on the [Poky](#) distribution which is a Linux distribution stripped down to a minimal configuration.

The provided distribution is based on [BusyBox](#) and built using [musl libc](#).

bbappend	<code>\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-kernel/linux/linux-yocto%.bbappend</code>
Recipe	<code>\$WORKSPACE/poky/meta/recipes-kernel/linux/linux-yocto_6.10.bb</code>
defconfig	<code>\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-kernel/linux/files/corstone1000/defconfig</code>

Secure Enclave Components

Trusted Firmware-M

bbappend	<code>\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-bsp/trusted-firmware-m/trusted-firmware-m%.bbappend</code>
Recipe	<code>\$WORKSPACE/meta-arm/meta-arm/recipes-bsp/trusted-firmware-m/trusted-firmware-m_2.1.0.bb</code>

External System Processor Components

RTX Real-Time operating system

An example application that uses the [RTX Real-Time Operating System](#).

The application project can be found [here](#).

Recipe	<code>\$WORKSPACE/meta-arm/meta-arm-bsp/recipes-bsp/external-system/external-system_0.1.0.bb</code>
--------	---

1.2.6 Build

Warning: Building binaries natively on Windows and AArch64 Linux is not supported.

Use an AMD64 Linux based development machine to build the software stack and transfer the binaries to run the software stack on an FVP in Windows or AArch64 Linux if required.

1. Create a new folder that will be your workspace.

```
mkdir $WORKSPACE
cd $WORKSPACE
```

2. Install kas version 4.4 with sudo rights.

```
sudo pip3 install kas==4.4
```

Ensure the kas installation directory is visible on the \$PATH environment variable.

3. Clone the *meta-arm* Yocto layer in the workspace \$WORKSPACE.

```
cd $WORKSPACE
git clone https://git.yoctoproject.org/git/meta-arm -b CORSTONE1000-2024.11
```

4. Build a Corstone-1000 image:

```
kas build meta-arm/kas/corstone1000-$TARGET.yml:meta-arm/ci/debug.yml
```

Important: Accept the EULA at <https://developer.arm.com/downloads/-/arm-ecosystem-fvps/eula> to build a Corstone-1000 image for FVP as follows:

```
export ARM_FVP_EULA_ACCEPT="True"
```

Warning: Access to the External System Processor is disabled by default. To build the Corstone-1000 image with External System Processor enabled, run:

```
kas build meta-arm/kas/corstone1000-$TARGET.yml:meta-arm/ci/debug.
↪ yml:meta-arm/kas/corstone1000-extsys.yml
```

A clean build takes a significant amount of time given that all of the development machine utilities are also built along with the target images. Those development machine utilities include executables (Python, CMake, etc.) and the required toolchains.

Once the build succeeds, all output binaries will be placed in \$WORKSPACE/build/tmp/ deploy/images/corstone1000-\$TARGET/

Everything apart from the Secure Enclave ROM firmware and External System firmware, is bundled into a single binary, the corstone1000-flash-firmware-image-corstone1000-\$TARGET.wic file.

The output binaries run in the Corstone-1000 platform are the following:

- The Secure Enclave ROM firmware: \$WORKSPACE/build/tmp/ deploy/images/corstone1000-\$TARGET/bl1.bin

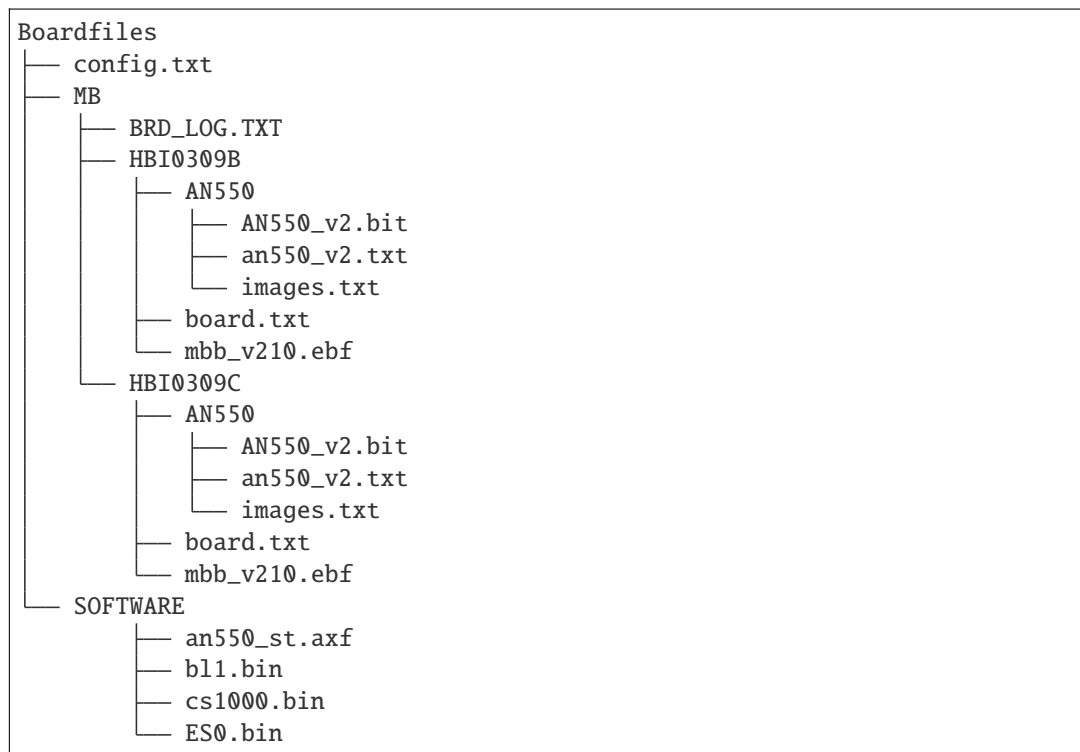
- The External System Processor firmware: `$WORKSPACE/build/tmp/deploy/images/corstone1000-$TARGET/es_flashfw.bin`
- The internal firmware flash image: `$WORKSPACE/build/tmp/deploy/images/corstone1000-$TARGET/corstone1000-flash-firmware-image-corstone1000-$TARGET.wic`

1.2.7 Flash

Note: The steps below only apply to the MPS3. The FVP being a software application running on your development machine does not require any firmware flashing. Refer to [this](#) section for running the software stack on FVP.

1. Download the FPGA bit file image AN550: Arm® Corstone™-1000 for MPS3 Version 2.0 on the [Arm Developer website](#). Click on the Download AN550 bundle button and login to download the file.

The directory structure of the FPGA bundle is as shown below:



2. Depending upon the MPS3 board version, you should update the `images.txt` file (found in the corresponding HBI0309x folder e.g. `Boardfiles/MB/HBI0309$BOARD_VERSION/AN550/images.txt`) so it points to the images under the `SOFTWARE` directory. Where `$BOARD_VERSION` is a variable containing the board printed on the MPS3 board.

The `images.txt` file compatible with the latest version of the software stack can be seen below;

```

; *****
;          Preload port mapping          *
; *****
; PORT 0 & ADDRESS: 0x00_0000_0000 QSPI Flash (XNVM) (32MB)
; PORT 0 & ADDRESS: 0x00_8000_0000 OCVN (DDR4 2GB)
; PORT 1          Secure Enclave (M0+) ROM (64KB)
  
```

(continues on next page)

(continued from previous page)

```
; PORT 2      External System 0 (M3) Code RAM (256KB)
; PORT 3      Secure Enclave OTP memory (8KB)
; PORT 4      CVM (4MB)
; *****
;

[IMAGES]
TOTALIMAGES: 3      ;Number of Images (Max: 32)

IMAGE0PORT: 1
IMAGE0ADDRESS: 0x00_0000_0000
IMAGE0UPDATE: RAM
IMAGE0FILE: \SOFTWARE\b11.bin

IMAGE1PORT: 0
IMAGE1ADDRESS: 0x00_0000_0000
IMAGE1UPDATE: AUTOQSPI
IMAGE1FILE: \SOFTWARE\cs1000.bin

IMAGE2PORT: 2
IMAGE2ADDRESS: 0x00_0000_0000
IMAGE2UPDATE: RAM
IMAGE2FILE: \SOFTWARE\es0.bin
```

3. Copy `b11.bin` from `$WORKSPACE/build/tmp/ deploy/images/corstone1000-mps3` to the `SOFTWARE` directory of the FPGA bundle.
4. Copy `es_flashfw.bin` from `$WORKSPACE/build/tmp/ deploy/images/corstone1000-mps3` to the `SOFTWARE` directory of the FPGA bundle and rename the binary to `es0.bin`.
5. Copy `corstone1000-flash-firmware-image-corstone1000-mps3.wic` from `$WORKSPACE/build/tmp/ deploy/images/corstone1000-mps3` to the `SOFTWARE` directory of the FPGA bundle and rename the wic image to `cs1000.bin`.

Note: Renaming of the images is required because the MCC firmware has a limit of 8 characters for file name and 3 characters for file extension.

After making all modifications above, copy the FPGA bit file bundle to the board's SDCard and reboot the MPS3.

1.2.8 Run

Once the target is turned ON, the Secure Enclave will start to boot, wherein the relevant memory contents of the *.wic file are copied to their respective memory locations. Firewall policies are enforced on memories and peripherals before bringing the Host Processor out of reset.

The Host Processor will boot TrustedFirmware-A, OP-TEE, U-Boot and then Linux before presenting a login prompt.

MPS3

1. Open 4 serial port comms terminals on the host machine. Those might be ttyUSB0, ttyUSB1, ttyUSB2, and ttyUSB3 on Linux machines.

- ttyUSB0 for MCC, OP-TEE and Secure Partition
- ttyUSB1 for Secure Enclave (Cortex-M0+)
- ttyUSB2 for Host Processor (Cortex-A35)
- ttyUSB3 for External System Processor (Cortex-M3)

The serial ports might be different on Windows machines.

Run the following commands in separate terminal instances on Linux:

```
sudo picocom -b 115200 /dev/ttyUSB0
```

```
sudo picocom -b 115200 /dev/ttyUSB1
```

```
sudo picocom -b 115200 /dev/ttyUSB2
```

```
sudo picocom -b 115200 /dev/ttyUSB3
```

Important: Plug a connected Ethernet cable to the MPS3 or it will wait for a network connection for a considerable amount of time, printing the following on the Host Processor terminal (ttyUSB2):

```
Generic PHY 40100000.ethernet-ffffffff:01: attached PHY driver (mii_bus:phy_
↪addr=40100000.ethernet-ffffffff:01, irq=POLL)
smsc911x 40100000.ethernet eth0: SMSC911x/921x identified at 0xfffffc008e50000,
↪IRQ: 17
Waiting up to 100 more seconds for network.
```

2. Once the system boot is completed, you should see console logs on the serial port terminals. Once the Host Processor is booted completely, user can login to the shell using root login.

Important: The secure flash might be completely filled if the system does not boot and only the Secure Enclave logs (ttyUSB1) are visible.

Clean the secure flash if that is the case following the steps [here](#).

FVP

A Fixed Virtual Platform (FVP) model of the Corstone-1000 platform must be available to run the Corstone-1000 FVP software image.

A Yocto recipe is provided to download the latest supported FVP version.

The recipe is located at `$WORKSPACE/meta-arm/meta-arm/recipes-devtools/fvp/fvp-corstone1000.bb`.

The latest FVP version is 11.23.25 and is automatically downloaded and installed when using the `runfvp` command as detailed below.

Note:

```
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml \  
-c "../meta-arm/scripts/runfvp -- --version"
```

The FVP can also be manually downloaded from the [Arm Ecosystem FVPs](#) page by navigating to “Corstone IoT FVPs” section to download the Corstone-1000 platform FVP installer. Follow the instructions of the installer to setup the FVP.

1. Run the FVP

```
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml \  
-c "../meta-arm/scripts/runfvp --terminals=tmux"
```

When the script is executed, three terminal instances will be launched:

- one for the Secure Enclave processing element
- two for the Host processor processing element.

```
corstone1000-fvp login:
```

2. Login using the root username.

1.2.9 Security Issue Reporting

To report any security issues identified with Corstone-1000, please send an email to psirt@arm.com.

1.3 Tests

Important: All the tests below assume you have already built the software stack at least once following the instructions [here](#).

1.3.1 Clean Secure Flash

Important: The MPS3 secure flash needs to be cleared before running tests. This is to erase the flash cleanly and prepare a clean board environment for testing.

1. Clone the *systemready-patch* repository to your \$WORKSPACE.

```
cd $WORKSPACE
git clone https://git.gitlab.arm.com/arm-reference-solutions/systemready-
↪patch.git -b CORSTONE1000-2024.11
```

2. Copy the secure flash cleaning Git patch file to your copy of *meta-arm*.

```
cp -f systemready-patch/embedded-a/corstone1000/erase_flash/0001-embedded-a-
↪corstone1000-clean-secure-flash.patch meta-arm
```

3. Apply the Git patch to *meta-arm*.

```
cd meta-arm
git apply 0001-embedded-a-corstone1000-clean-secure-flash.patch
```

4. Rebuild the software stack.

```
cd $WORKSPACE
kas shell meta-arm/kas/corstone1000-mps3.yml:meta-arm/ci/debug.yml
bitbake -c cleansstate trusted-firmware-m corstone1000-flash-firmware-image
bitbake -c build corstone1000-flash-firmware-image
```

5. Replace the *b11.bin* file on the SD card with *\$WORKSPACE/build/tmp/ deploy/images/corstone1000-mps3/b11.bin*.

6. Reboot the board to completely erase the secure flash.

The following message log from TrustedFirmware-M should be displayed on the Secure Enclave terminal (ttyUSB1):

```
!!!SECURE FLASH HAS BEEN CLEANED!!!
NOW YOU CAN FLASH THE ACTUAL CORSTONE1000 IMAGE
PLEASE REMOVE THE LATEST ERASE SECURE FLASH PATCH AND BUILD THE IMAGE AGAIN
```

7. Whilst still in the *kas* shell, revert the changes the patch introduced by running the following commands:

```
cd $WORKSPACE/meta-arm
git reset --hard
cd ..
bitbake -c cleansstate trusted-firmware-m corstone1000-flash-firmware-image
exit
```

8. Follow the *instructions* to build a clean software stack and flash the MPS3 with it.

You can proceed with the test instructions in the following section after having done all the above.

1.3.2 SystemReady-IR

Important: Running the SystemReady-IR tests described below requires USB drives. In our testing, not all USB drive models worked well with the MPS3.

Here are the USB drive models that were stable in our test environment:

- HP v165w 8 GB USB Flash Drive
- SanDisk Ultra 32GB Dual USB Flash Drive USB M3.0
- SanDisk Ultra 16GB Dual USB Flash Drive USB M3.0

Follow the instructions below before running the Architecture Compliance Suite (ACS) tests.

Build an EFI System Partition

A storage with EFI System Partition (ESP) must exist in the system for the UEFI-SCT related tests to pass.

1. Build an ESP partition for your target

```
kas build meta-arm/kas/corstone1000-$TARGET.yml:meta-arm/ci/debug.yml --  
↪target corstone1000-esp-image
```

2. Locate the `corstone1000-esp-image-corstone1000-$TARGET.wic` build artefact in `$WORKSPACE/build/tmp/deploy/images/corstone1000-$TARGET/`

Use the EFI System Partition

MPS3

1. Connect a USB drive to your development machine.
2. Run the following command on your development machine to discover which device is your USB drive:

```
lsblk
```

The remaining steps assume the USB drive is `/dev/sdb`.

Warning: Do not mistake your development machine hard drive with the USB drive.

3. Copy the ESP to the USB drive by running the following command:

```
sudo dd \  
if=$WORKSPACE/build/tmp/deploy/images/corstone1000-mps3/corstone1000-esp-  
↪image-corstone1000-mps3.wic \  
of=/dev/sdb \  
iflag=direct oflag=direct status=progress bs=512; sync;
```

4. Plug the USB drive to the MPS3.

FVP

The ESP disk image will automatically be used by the Corstone-1000 FVP as the 2nd MMC card image. It will be used when the SystemReady-IR tests is performed on the FVP in the later section.

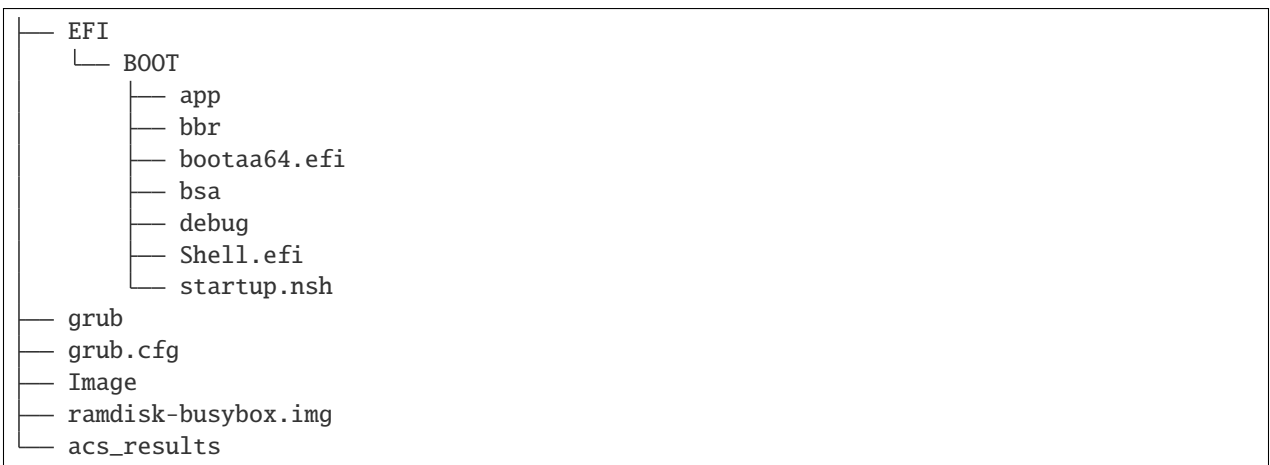
Run SystemReady-IR ACS Tests

ACS is used to ensure architectural compliance across different implementations of the architecture. Arm Enterprise ACS includes a set of examples of the invariant behaviors that are provided by a set of specifications for enterprise systems (i.e. SBSA, SBBR, etc.). Implementers can verify if these behaviors have been interpreted correctly.

The following test suites and bootable applications are under the BOOT partition of the ACS image:

- SCT
- FWTS
- BSA UEFI
- BSA linux
- GRUB
- UEFI manual capsule application

See the directory structure of the ACS image BOOT partition below:



The BOOT partition is also used to store test results in the `acs_results` folder.

Important: Ensure that the `acs_results` folder is empty before starting the test.

This sections below describe how to build and run ACS tests on Corstone-1000.

1. On your host development machine, clone the [Arm SystemReady ACS repository](https://github.com/ARM-software/arm-systemready).

```
cd $WORKSPACE
git clone https://github.com/ARM-software/arm-systemready.git
```

This repository contains the infrastructure to build the ACS and the bootable prebuilt images to be used for the certifications of SystemReady-IR.

2. Find the pre-built ACS live image in `$WORKSPACE/arm-systemready/IR/prebuilt_images/v23.09_2.1.0/ir-acs-live-image-generic-arm64.wic.xz`.

Note: This prebuilt ACS image includes v5.13 kernel, which does not provide USB driver support for Corstone-1000. The ACS image with a newer kernel version and full USB support for Corstone-1000 will be available in the repository with the next SystemReady release.

3. Decompress the pre-built ACS live image.

```
cd $WORKSPACE/arm-systemready/IR/prebuilt_images/v23.09_2.1.0
unxz ir-acs-live-image-generic-arm64.wic.xz
```

MPS3

1. Connect a USB drive (other than the one used for the ESP) to the host development machine.
2. Run the following command to discover which device is your USB drive:

```
lsblk
```

The remaining steps assume the USB drive is `/dev/sdc`.

Warning: Do not mistake your development machine hard drive with the USB drive.

3. Copy the ACS image to the USB drive by running the following commands:

```
cd $WORKSPACE/arm-systemready/IR/prebuilt_images/v23.09_2.1.0
sudo dd if=ir-acs-live-image-generic-arm64.wic of=/dev/sdc iflag=direct,
↳ oflag=direct bs=1M status=progress; sync
```

4. Plug the USB drive to the MPS3. At this point you should have both the USB drive with the ESP and the USB drive with the ACS image plugged to the MPS3.
5. Reboot the MPS3.

The MPS3 will reset multiple times during the test, and it might take approximately 24 to 36 hours to finish the test.

Important: Unplug the ESP USB drive from the MPS3 if it is preventing GRUB from finding the bootable partition. Leave only the ACS image USB drive plugged in to run the ACS tests.

The ESP USB drive can be plugged in again after selecting the *Linux Boot* option in the GRUB menu at the end of the ACS tests.

Warning: A timeout issue has been observed while booting Linux during the ACS tests, causing the system to boot into emergency mode. Booting Linux is necessary to run certain tests, such as *dt-validation*. The following workaround is required to enable Linux to boot properly and perform all Linux-based tests:

1. Press Enter at the Linux prompt.
2. Open the file `/etc/systemd/system.conf` and set `DefaultDeviceTimeoutSec=infinity`.

3. Reboot the platform using the *reboot* command.
4. Select the *Linux Boot* option from the GRUB menu.
5. Allow Linux to boot and run the remaining ACS tests until completion.

FVP

Run the commands below to run the ACS test on FVP using the built firmware image and the pre-built ACS image identified above:

```
cd $WORKSPACE
tmux
./meta-arm/scripts/runfvp \
--terminals=tmux \
./build/tmp/deploy/images/corstone1000-fvp/corstone1000-flash-firmware-image-
↪corstone1000-fvp.fvpconf \
-- -C board.msdcmmc.p_mmc_file=$WORKSPACE/arm-systemready/IR/prebuilt_images/v23.09_2.1.
↪0/ir-acs-live-image-generic-arm64.wic
```

Note: The FVP will reset multiple times during the test. The ACS tests might take up to 1 day to complete when run on FVP.

The message *ACS run is completed* will be displayed on the FVP host terminal when the test runs to completion. You will be prompted to press the Enter key to access the Linux prompt.

Test Sequence and Results

U-Boot should be able to boot the GRUB bootloader from the first partition.

If GRUB is not interrupted, the tests are executed automatically in the following order:

- SCT
- UEFI BSA
- FWTS

The results can be fetched from the *acs_results* folder in the BOOT partition of the USB drive (for MPS3) or SD Card (for FVP).

Note: Access the *acs_results* folder in FVP by running the following commands:

```
sudo mkdir /mnt/test
sudo mount -o rw,offset=1048576 \
$WORKSPACE/arm-systemready/IR/prebuilt_images/v23.09_2.1.0/ir-acs-live-image-generic-
↪arm64.wic \
/mnt/test
```

1.3.3 Capsule Update

The following section describes the steps to update the firmware using Capsule Update as the Corstone-1000 supports UEFI.

The firmware update process is tested with an invalid capsule (negative capsule update test) and with a valid capsule (positive capsule update test) to validate the robustness and error-handling capabilities of the firmware update mechanism.

During the positive capsule update test, the Corstone-1000 is given a valid capsule, which it successfully applies, boots up and then reaches the Linux command prompt.

During the negative capsule update test, the Corstone-1000 is given an outdated capsule with a lower version number, which is expected to be rejected due to its outdated status, thereby retaining the previous firmware.

Two different capsules (one for each test) are therefore needed to perform the tests.

Generate Capsules

U-Boot's `mkeficap` tool is used to generate capsules. It is built automatically for the host machine during the firmware image building process. The tool can be found in the `$WORKSPACE/build/tmp/sysroots-components/x86_64/u-boot-tools-native/usr/bin/mkeficap` directory.

`mkeficap` uses a no-partition image which is created when performing a clean firmware build. The no-partition image can be found in the `$WORKSPACE/build/tmp/deploy/images/corstone1000-$TARGET/corstone1000-$TARGET_image.nopt` directory.

The capsule's default metadata passed can be found in the `$WORKSPACE/meta-arm/meta-arm-bsp/recipes-bsp/images/corstone1000-flash-firmware-image.bb` and `$WORKSPACE/meta-arm/kas/corstone1000-image-configuration.yml` files.

Valid Capsule

An automatically generated capsule can be found in `$WORKSPACE/build/tmp/deploy/images/corstone1000-$TARGET/corstone1000-$TARGET-v6.uefi.capsule` after running a firmware build.

The default metadata values are assumed to be correct to generate a valid capsule.

This capsule will be used for the positive capsule update test.

Invalid Capsule

Generate another capsule with `fw-version` metadata set to a lower version than the valid capsule. The example below assumes the valid capsule has a default firmware version of 6, and therefore creates an invalid capsule with firmware version 5.

Run the following commands to generate an invalid capsule with a `fw-version` of 5:

```
cd $WORKSPACE

./build/tmp/sysroots-components/x86_64/u-boot-tools-native/usr/bin/mkeficap \
--monotonic-count 1 \
--private-key build/tmp/deploy/images/corstone1000-$TARGET/corstone1000_capsule_key.key \
--certificate build/tmp/deploy/images/corstone1000-$TARGET/corstone1000_capsule_cert.crt
↵
```

(continues on next page)

(continued from previous page)

```
--index 1 \
--guid $TARGET_GUID \
--fw-version 5 build/tmp/deploy/images/corstone1000-$TARGET/corstone1000-$TARGET_image.
↪nopt \
corstone1000-$TARGET-v5.uefi.capsule
```

Important: \$TARGET_GUID is different depending on whether the capsule is built for the fvp or mps3 \$TARGET.

- fvp \$TARGET_GUID is 989f3a4e-46e0-4cd0-9877-a25c70c01329
- mps3 \$TARGET_GUID is df1865d1-90fb-4d59-9c38-c9f2c1bba8cc

The invalid capsule will be located in the \$WORKSPACE directory.

Transfer Capsules to Target

The capsule delivery process described below is the direct method (usage of capsules from the ACS image) as opposed to the on-disk method (delivery of capsules using a file on a mass storage device).

MPS3

1. Prepare a USB drive as explained in *this* section.
2. Copy the capsule file to the root directory of the BOOT partition in the USB drive.

```
sudo cp $CAPSULES_PATH/corstone1000-mps3-v6.uefi.capsule $ACS_IMAGE_USB_DRIVE_
↪PATH/BOOT/
sudo cp $CAPSULES_PATH/corstone1000-mps3-v5.uefi.capsule $ACS_IMAGE_USB_DRIVE_
↪PATH/BOOT/
sync
```

Important: Since we are using the direct Capsule Update method, the capsule files should not be placed in the EFI/UpdateCapsule directory, as this might inadvertently trigger the on-disk update method.

FVP

1. Download and extract the ACS image *as described for the MPS3*. The ACS image extraction location will be referred below as \$ACS_IMAGE_PATH.

Note: Creating a USB drive with the ACS image is not required as the image will be mounted with the steps below.

2. Find the first partition's offset of the `ir-acs-live-image-generic-arm64.wic` image using the `fdisk` tool. The partition table can be listed using:

```
fdisk -lu $ACS_IMAGE_PATH/ir-acs-live-image-generic-arm64.wic
Device                               Start      End
↳ Sectors  Size Type
$ACS_IMAGE_PATH/ir-acs-live-image-generic-arm64.wic1    2048  309247
↳ 307200  150M Microsoft basic data
$ACS_IMAGE_PATH/ir-acs-live-image-generic-arm64.wic2  309248 1343339
↳ 1034092  505M Linux filesystem
```

Given that the first partition starts at sector 2048 and each sector is 512 bytes in size, the first partition is at offset 1048576 (2048 x 512).

3. Mount the `ir-acs-live-image-generic-arm64.wic` image using the previously calculated offset:

```
sudo mkdir /mnt/ir-acs-live-image-generic-arm64
sudo mount -o rw,offset=<first_partition_offset> $ACS_IMAGE_PATH/ir-acs-
↳ live-image-generic-arm64.wic /mnt/ir-acs-live-image-generic-arm64
```

4. Copy the capsules:

```
sudo cp $CAPSULES_PATH/corstone1000-fvp-v6.uefi.capsule /mnt/ir-acs-live-
↳ image-generic-arm64/
sudo cp $CAPSULES_PATH/corstone1000-fvp-v5.uefi.capsule /mnt/ir-acs-live-
↳ image-generic-arm64/
sync
```

5. Unmount the IR image:

```
sudo umount /mnt/ir-acs-live-image-generic-arm64
```

Run Capsule Update Tests

The valid capsule (`corstone1000-$TARGET-v6.uefi.capsule`) will be used first to run the positive capsule update test. This will be followed by using the invalid capsule (`corstone1000-$TARGET-v5.uefi.capsule`) to run the negative capsule update test.

Important: This sequence order must be respected as the invalid capsule has a firmware version lower than the firmware version in the valid capsule. The negative capsule update test effectively tests that firmware rollback is not permitted.

Positive Capsule Update Test

1. Run Corstone-1000 with the ACS image containing the two capsule files:
 - MPS3:
 1. Plug the prepared USB drive which has the IR prebuilt image and two capsules to the MPS3.
 2. Power cycle the MPS3.
 - FVP:
 1. Run the FVP with the IR prebuilt image which now also contains the two capsules:

```
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml \
-c "../meta-arm/scripts/runfvp --terminals=tmux \
-- -C board.msd_mmc.p_mmc_file=$ACS_IMAGE_PATH/ir-ac-live-image-generic-arm64.
↵wic"
```

Warning: \$ACS_IMAGE_PATH must be an absolute path. Ensure there are no spaces before or after of = of the -C board.msd_mmc.p_mmc_file option.

- Wait until U-Boot loads EFI from the ACS image and interrupt the EFI shell by pressing the Escape key when the following prompt is displayed on the Host Processor terminal (ttyUSB2).

```
Press ESC in 4 seconds to skip startup.nsh or any other key to continue.
```

- Access the content of the first file system (File System 0) where we copied the capsule files by running the following command:

```
FS0:
```

- Run the CapsuleApp application with the valid capsule file:

- MPS3:

```
EFI/BOOT/app/CapsuleApp.efi EFI/BOOT/corstone1000-mps3-v6.uefi.
↵capsule
```

- FVP:

```
EFI/BOOT/app/CapsuleApp.efi corstone1000-fvp-v6.uefi.capsule
```

The capsule update will be started.

Note: The capsule update takes about 8 minutes to complete on MPS3 and between 15-30 minutes on FVP.

The Corstone-1000 will reset after successfully applying the capsule.

The software stack copies the capsule content to the external flash, which is shared between the Secure Enclave and the Host Processor before rebooting the system.

After the first reboot, TrustedFirmware-M should apply the valid capsule and display the following log on the Secure Enclave terminal (ttyUSB1) before rebooting the system a second time:

```
...
SysTick_Handler: counted = 10, expiring on = 360
SysTick_Handler: counted = 20, expiring on = 360
SysTick_Handler: counted = 30, expiring on = 360
...
metadata_write: success: active = 1, previous = 0
flash_full_capsule: exit
corstone1000_fwu_flash_image: exit: ret = 0
...
```

The above log snippet indicates that the new capsule image is successfully applied, and the board is booting with the external flash's Bank-1.

After a second reboot, the following log should be displayed on on the Secure Enclave terminal (ttyUSB1):

```
...
fmp_set_image_info:133 Enter
FMP image update: image id = 0
FMP image update: status = 0version=6 last_attempt_version=6.
fmp_set_image_info:157 Exit.
corstone1000_fwu_host_ack: exit: ret = 0
...
```

5. Interrupt the U-Boot shell.

```
Hit any key to stop autoboot:
```

6. Run the following commands in order to run the Corstone-1000 Linux kernel.

Note: Otherwise, the execution ends up in the ACS live image.

```
$ unzip $kernel_addr 0x90000000
$ loadm 0x90000000 $kernel_addr_r $filesize
$ bootefi $kernel_addr_r $fdtcontroladdr
```

7. After the system fully boots, read the EFI System Resource Table (ESRT) to verify that the firmware version matches the version of the capsule applied.

```
# cd /sys/firmware/efi/esrt/entries/entry0
# cat *

0x0                                # capsule_flags
989f3a4e-46e0-4cd0-9877-a25c70c01329 # fw_class
0                                   # fw_type
6                                   # fw_version
0                                   # last_attempt_status
6                                   # last_attempt_version
0                                   # lowest_supported_fw_ver
```

See the [UEFI documentation](#) for more information on the significance of the table fields.

Warning: Do not terminate FVP between the positive and negative capsule update tests.

Negative Capsule Update Test

Important: The *positive capsule update test* must be run before running the negative capsule update test.

1. After running the positive capsule update test, reboot the system by typing the following command on the Host Processor terminal (ttyUSB2):

```
reboot
```

2. Wait until U-Boot loads EFI from the ACS image and interrupt the EFI shell by pressing the Escape key when the following prompt is displayed on the Host Processor terminal (ttyUSB2).

```
Press ESC in 4 seconds to skip startup.nsh or any other key to continue.
```

3. Access the content of the first file system (File System 0) where we copied the capsule files by running the following command:

```
FS0:
```

4. Run the CapsuleApp application with the invalid capsule file:

- MPS3:

```
EFI/BOOT/app/CapsuleApp.efi EFI/BOOT/corstone1000-mps3-v5.uefi.capsule
```

- FVP:

```
EFI/BOOT/app/CapsuleApp.efi corstone1000-fvp-v5.uefi.capsule
```

5. TrustedFirmware-M should reject the capsule due to having a lower firmware version and display the following log on the Secure Enclave terminal (ttyUSB1):

```
...
uefi_capsule_retrieve_images: image 0 at 0xa0000070, size=15654928
uefi_capsule_retrieve_images: exit
flash_full_capsule: enter: image = 0x0xa0000070, size = 7764541, version
↵= 5
ERROR: flash_full_capsule: version error
private_metadata_write: enter: boot_index = 1
private_metadata_write: success
fmp_set_image_info:133 Enter
FMP image update: image id = 0
FMP image update: status = 1version=6 last_attempt_version=5.
fmp_set_image_info:157 Exit.
corstone1000_fwu_flash_image: exit: ret = -1
fmp_get_image_info:232 Enter
pack_image_info:207 ImageInfo size = 105, ImageName size = 34,
↵ImageVersionName
size = 36
fmp_get_image_info:236 Exit
...
```

The Secure Enclave tries to load the new image a predetermined number of times if the capsule passes initial verification but fails verifications performed during boot time.

```

...
metadata_write: success: active = 0, previous = 1
fwu_select_previous: in regular state by choosing previous active_
↪bank
...

```

The Secure Enclave eventually reverts back to the previously running image.

6. Reboot manually:

```
Shell> reset
```

7. Interrupt the U-Boot shell.

```
Hit any key to stop autoboot:
```

8. Run the following commands in order to run the Corstone-1000 Linux kernel.

Note: Otherwise, the execution ends up in the ACS live image.

```

$ unzip $kernel_addr 0x90000000
$ loadm 0x90000000 $kernel_addr_r $filesize
$ bootefi $kernel_addr_r $fdtcontroladdr

```

9. After the system fully boots, read the ESRT to verify the firmware version does not match what is on the invalid capsule.

```

# cd /sys/firmware/efi/esrt/entries/entry0
# cat *

0x0                # capsule_flags
989f3a4e-46e0-4cd0-9877-a25c70c01329 # fw_class
0                  # fw_type
6                  # fw_version
1                  # last_attempt_status
5                  # last_attempt_version
0                  # lowest_supported_fw_ver

```

1.3.4 Linux Distributions

This sections describes the steps to install major Linux distributions to the Corstone-1000 Host Processor.

The Linux distributions to be installed are:

- Debian
- openSUSE

Follow the instructions below to install the Linux distributions to the Corstone-1000 software stack.

Prepare Installation Media

The media containing the bootable files required to start the installation process needs to be prepared.

Follow the instructions below to create the installation media.

- Using your development machine, download one of following Linux distribution images:

- [Debian installer image](#)
- [OpenSUSE Tumbleweed installer image](#)

Note: For openSUSE Tumbleweed, search for an ISO file with the format: `openSUSE-Tumbleweed-DVD-aarch64-Snapshot$DATE-Media.iso`.

`openSUSE-Tumbleweed-DVD-aarch64-Snapshot20240516-Media.iso` was used during development.

The location of the ISO file on the development machine will be referred to as `$DISTRO_INSTALLER_ISO_PATH`.

- Create the installation media which will contain the necessary files to install the operation system.
 - MPS3:
 - Plug a blank USB drive formatted with FAT32, ensuring it has a minimum capacity of 4GB, to the development machine.
 - Run the following command to discover which device is your USB drive:

```
lsblk
```

The remaining steps assume the USB drive is `/dev/sdb`.

Warning: Do not mistake your development machine hard drive with the USB drive.

- Write one of the distribution installer ISO file to the USB drive.

```
sudo dd if=$DISTRO_INSTALLER_ISO_PATH of=/dev/sdb iflag=direct,
↳ oflag=direct status=progress bs=1M; sync;
```

- FVP:

The distribution installer ISO file does not need to be burnt to a USB drive. It will be used as is when starting the FVP install the distribution.

Prepare System Drive

A system (or boot) drive, to store all the operating system files and used to boot the distribution, is required as Corstone-1000 on-board non-volatile storage size is insufficient for installing the distributions.

- MPS3:
 - Find another blank USB drive formatted with FAT32 with a minimum capacity of 4GB.
 - Do not yet connect this blank USB drive to the MPS3. It will be used as the primary drive to boot the distribution.

• **FVP:**

1. Create an 10 GB GUID Partition Table (GPT) formatted MultiMediaCard (MMC) image.

```
dd if=/dev/zero of=$WORKSPACE/fvp_distro_system_drive.img \  
bs=1 count=0 seek=10G; sync; \  
parted -s fvp_distro_system_drive.img mklabel gpt
```

2. This MMC image will be used as the primary drive to boot the distribution.

Installation

MPS3

1. Connect the installation media, which contains the installer for the desired distribution, to the MPS3.
2. Open a serial port terminal interface to `/dev/ttyUSB0` in one terminal window on your development machine.

```
sudo picocom -b 115200 /dev/ttyUSB0
```

3. Open a serial port terminal interface to `/dev/ttyUSB2` in another terminal window on your development machine.

```
sudo picocom -b 115200 /dev/ttyUSB2
```

4. When the installation screen is displayed on `ttyUSB2`, plug in the (still empty) system drive to the MPS3.
5. Start the distribution installation process.

Note: Reboot the MPS3 with both USB drives (installation media and empty system drive) connected to it if the distribution installer does not start.

Note: Due to the performance limitation, the distribution installation process can take up to 24 hours to complete.

FVP

1. Start the FVP with the system drive as the primary drive and the distro ISO file as the secondary drive.

```
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml \  
-c "../meta-arm/scripts/runfvp --terminals=tmux -- \  
-C board.msdc_mmc.p_mmc_file=$WORKSPACE/fvp_distro_system_drive.img \  
-C board.msdc_mmc_2.p_mmc_file=$DISTRO_INSTALLER_ISO_PATH"
```

The Linux distribution will be installed on `fvp_distro_system_drive.img`.

Debian Installation Extra Steps

Debian installation may need some extra steps, that are indicated below:

1. Answer **Yes** to the question **Force grub installation to the EFI removable media path?**.

If the GRUB installation fails, these are the steps to follow on the subsequent popups:

1. Select **Continue**, then **Continue** again on the next popup.
2. Scroll down and select **Execute a shell**.
3. Select **Continue**.
4. Enter the following command:

```
in-target grub-install --no-nvram --force-extra-removable
```

5. Enter the following command:

```
in-target update-grub
```

6. Enter the following command:

```
exit
```

7. Select **Continue without boot loader**, then select **Continue** on the next popup.
8. At this stage, the installation should proceed as normal.

2. Answer **No** to the question **Update NVRAM variables to automatically boot into Debian?**.

Boot Distribution

- MPS3

1. Once the installation is complete, unplug the installation media.
2. Perform a cold boot of the MPS3.

- FVP

The target should automatically boot into the installed operating system image.

Stop the FVP and run the command below to simulate a cold boot:

```
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml \
-c "../meta-arm/scripts/runfvp --terminals=tmux -- \
-C board.msd_mmc.p_mmc_file=$WORKSPACE/fvp_distro_system_drive.img"
```

Warning: To manually enter recovery mode, once the FVP begins booting, you can quickly change the boot option in GRUB, to boot into recovery mode. This option will disappear quickly, so it is best to preempt it.

Select **Advanced Options** for <OS> and then <OS> (recovery mode).

The target will then enter recovery mode, from which the user can access a shell after entering the password for the root user.

Timeout Optimizations

Important: Operating system timeouts are inconsistent across systems. Skip this section if the system boots to Debian or OpenSUSE without any issue.

Make the system modification below whilst in recovery mode to increase timeouts and boot to the installed distribution.

1. Remove the timeout limit for device operations.

- **Debian**

```
vi /etc/systemd/system.conf
DefaultDeviceTimeoutSec=infinity
```

- **openSUSE**

```
vi /usr/lib/systemd/system.conf
DefaultDeviceTimeoutSec=infinity
```

Warning: As modifying `system.conf` in `/usr/lib/systemd/` is not working as it is getting overwritten, copy `system.conf` from `/usr/lib/systemd/` to `/etc/systemd/system.conf`.
d/ after the above edit.

2. Set the maximum time that the system will wait for a user to successfully log in before timing out to 180 seconds.

- **Debian**

```
vi /etc/login.defs
LOGIN_TIMEOUT 180
```

- **openSUSE**

```
vi /usr/etc/login.defs
LOGIN_TIMEOUT 180
```

3. Ensure the changes are applied by run the command below.

```
systemctl daemon-reload
```

4. Perform a cold boot of the target.

Log into the Distribution

Login with the `root` username and its corresponding password (set during installation) at the distribution login prompt after booting. See an illustration for Debian below:

```
debian login:
```

1.3.5 UEFI Secure Boot

The UEFI Secure Boot test is designed to verify the integrity and authenticity of the system's boot process. This test ensures that only trusted, signed images are executed, thereby preventing unauthorized or malicious code from running. A successful test confirms that the signed image executes correctly, while any unsigned image is blocked from running.

Generate Keys, Signed Image and Unsigned Image

1. Build an EFI System Partition as described [here](#).
2. Clone the *systemready-patch* repository to your workspace.

```
cd $WORKSPACE

git clone https://git.gitlab.arm.com/arm-reference-solutions/systemready-
↪patch.git \
-b CORSTONE1000-2024.11
```

3. Set the current working directory to build directory's subdirectory containing the software stack build images.

```
cd $WORKSPACE/build/tmp/deploy/images/corstone1000-$TARGET/
```

4. Run the image signing script (without changing the current working directory).

```
./$WORKSPACE/systemready-patch/embedded-a/corstone1000/secureboot/create_
↪keys_and_sign.sh \
-d $TARGET \
-v $CERTIFICATE_VALIDITY_DURATION_IN_DAYS
```

Important: The *efitools* package is required to execute the script.

Note: Consult the image signing script help message (-h) for more information about other optional arguments.

The script is interactive and contains commands that require `sudo` level permissions.

The keys, signed kernel image, and unsigned kernel image will be copied to the existing ESP image. The modified ESP image can be found at `$WORKSPACE/build/tmp/deploy/images/corstone1000-$TARGET/corstone1000-esp-image-corstone1000-$TARGET.wic`.

Run Unsigned Image Boot Test

FVP

1. Follow the instructions [here](#) to use the ESP.
2. Run the software stack as described [here](#).
3. On the Host Processor terminal host side, stop the execution of U-Boot when prompted to do so with the message `Press any key to stop`.

Warning: There is a timeout of 3 seconds to stop the execution at the U-Boot prompt.

The U-Boot console prompt looks as follows:

```
corstone1000#
```

Important: The rest of the instructions below will be executed on the U-Boot terminal.

4. On the U-Boot console, set the current MMC device.

```
corstone1000# mmc dev 1
```

5. Enroll the four UEFI secure boot authenticated variables.

```
corstone1000# \  
load mmc 1:1 $loadaddr corstone1000_secureboot_keys/PK.auth && setenv -e -  
↪nv -bs -rt -at -i $loadaddr:$filesize PK; \  
load mmc 1:1 $loadaddr corstone1000_secureboot_keys/KEK.auth && setenv -e -  
↪nv -bs -rt -at -i $loadaddr:$filesize KEK; \  
load mmc 1:1 $loadaddr corstone1000_secureboot_keys/db.auth && setenv -e -  
↪nv -bs -rt -at -i $loadaddr:$filesize db; \  
load mmc 1:1 $loadaddr corstone1000_secureboot_keys/dbx.auth && setenv -e -  
↪nv -bs -rt -at -i $loadaddr:$filesize dbx
```

6. Attempt to Load the unsigned kernel image.

```
corstone1000# \  
load mmc 1:1 $loadaddr corstone1000_secureboot_fvp_images/Image_fvp; \  
loadm $loadaddr $kernel_addr_r $filesize; \  
bootefi $kernel_addr_r $fdtcontroladdr  
  
Booting /MemoryMapped(0x0,0x88200000,0x236aa00)  
Image not authenticated  
Loading image failed
```

The unsigned Linux kernel image should not be loaded.

MPS3

1. Follow the instructions [here](#) to use the ESP.
2. Perform a cold boot of the MPS3.
3. On the Host Processor terminal host side, stop the execution of U-Boot when prompted to do so with the message Press any key to stop.

Warning: There is a timeout of 3 seconds to stop the execution at the U-Boot prompt.

The U-Boot console prompt looks as follows:

```
corstone1000#
```

Important: The rest of the instructions below will be executed on the U-Boot terminal.

4. On the U-Boot console, reset USB.

```
corstone1000# usb reset
resetting USB...
Bus usb@40200000: isp1763 bus width: 16, oc: not available
USB ISP 1763 HW rev. 32 started
scanning bus usb@40200000 for devices... port 1 high speed
3 USB Device(s) found
    scanning usb for storage devices... 1 Storage Device(s) found
```

Note: Occasionally, the USB reset may fail to detect the USB device. It is advisable to rerun the USB reset command.

5. Select the first USB device, which should be the USB drive containing the ESP.

```
corstone1000# usb dev 0
```

6. Enroll the four UEFI secure boot authenticated variables.

```
corstone1000# \
load usb 0 $loadaddr corstone1000_secureboot_keys/PK.auth && setenv -e -nv -
↳-bs -rt -at -i $loadaddr:$filesize PK; \
load usb 0 $loadaddr corstone1000_secureboot_keys/KEK.auth && setenv -e -nv
↳-bs -rt -at -i $loadaddr:$filesize KEK; \
load usb 0 $loadaddr corstone1000_secureboot_keys/db.auth && setenv -e -nv -
↳-bs -rt -at -i $loadaddr:$filesize db; \
load usb 0 $loadaddr corstone1000_secureboot_keys/dbx.auth && setenv -e -nv
↳-bs -rt -at -i $loadaddr:$filesize dbx
```

7. Attempt to Load the unsigned kernel image.

```
corstone1000# \
load usb 0 $loadaddr corstone1000_secureboot_mps3_images/Image_mps3
loadm $loadaddr $kernel_addr_r $filesize
bootefi $kernel_addr_r $fdtcontroladdr

Booting /MemoryMapped(0x0,0x88200000,0x236aa00)
Image not authenticated
Loading image failed
```

The unsigned Linux kernel image should not be loaded.

Run Signed Image Boot Test

FVP

Important: You must first perform the *Unsigned Image Boot Test*.

Load the signed kernel image.

```
corstone1000# \  
load mmc 1:1 $loadaddr corstone1000_secureboot_fvp_images/Image_fvp.signed; \  
loadm $loadaddr $kernel_addr_r $filesize; \  
bootefi $kernel_addr_r $fdtcontroladdr
```

The signed Linux kernel image should be booted successfully.

MPS3

Important: You must first perform the *Unsigned Image Boot Test*.

Load the signed kernel image.

```
corstone1000# \  
load usb 0 $loadaddr corstone1000_secureboot_mps3_images/Image_mps3.signed; \  
loadm $loadaddr $kernel_addr_r $filesize; \  
bootefi $kernel_addr_r $fdtcontroladdr
```

The signed Linux kernel image should be booted successfully.

Disable Secure Boot

Running the UEFI Secure Boot Test steps stores UEFI authenticated variables in the secure flash. As a result, U-Boot reads these variables and verifies the Linux kernel image before executing it at each reboot.

In a typical boot scenario, the Linux kernel image is not signed, which will prevent the system from booting due to failed image authentication. To resolve this, the Platform Key (one of the UEFI authenticated variables for secure boot) needs to be deleted.

1. Perform a cold boot of the MPS3.
2. On the Host Processor terminal host side, stop the execution of U-Boot when prompted to do so with the message `Press any key to stop`.
3. On the U-Boot console, delete the Platform Key (PK).
 - FVP

```
corstone1000# \  
mmc dev 1; \  
load mmc 1:1 $loadaddr corstone1000_secureboot_keys/PK_delete.auth && \  
↪setenv -e -nv -bs -rt -at -i $loadaddr:$filesize PK; \  
boot
```

- MPS3

```
corstone1000# \
usb reset; \
usb dev 0; \
load usb 0 $loadaddr corstone1000_secureboot_keys/PK_delete.auth && \
↪setenv -e -nv -bs -rt -at -i $loadaddr:$filesize PK; \
boot
```

1.3.6 PSA API

The following tests the implementation of the Application Programming Interface (API) of the Platform Security Architecture (PSA) certification scheme. It uses Arm Firmware Framework for Arm A-profile (FF-A) to communicate between the normal world and the secure world to run the [Arm Platform Security Architecture Test Suite](#).

The tests use the `arm_tstee` driver to access Trusted Services Secure Partitions from user space. The driver is included in the Linux Kernel, starting from v6.10.

Important: Ensure there are no USB drives connected to the board when running the test on the MPS3.

The steps below are applicable to both MPS3 and FVP).

1. Start the Corstone-1000 and wait until it boots to Linux on the Host Processor terminal (ttyUSB2).
2. Run the PSA API tests by running the commands below in the order shown:

```
psa-iat-api-test
psa-crypto-api-test
psa-its-api-test
psa-ps-api-test
```

1.3.7 External System Processor

Important: Access to the External System Processor is disabled by default. Ensure you are running a software stack image with access to the External System Processor enabled following the steps [here](#).

The Linux operating system running on the Host Processor starts the `remoteproc` framework to manage the External System Processor.

1. Stop the External System Processor with the following command:

```
echo stop > /sys/class/remoteproc/remoteproc0/state
```

2. Start the External System Processor with the following command:

```
echo start > /sys/class/remoteproc/remoteproc0/state
```

1.3.8 Symmetric Multiprocessing

Warning: Symmetric multiprocessing (SMP) mode is only supported on FVP but is disabled by default.

1. Build the software stack with SMP mode enabled:

```
kas build meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml:meta-arm/
↳kas/corstone1000-fvp-multicore.yml
```

2. Run the Corstone-1000 FVP:

```
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml:meta-arm/
↳kas/corstone1000-fvp-multicore.yml \
-c "../meta-arm/scripts/runfvp"
```

3. Verify that the FVP is running the Host Processor with more than one CPU core:

```
nproc
4                # number of processing units
```

1.3.9 Secure Debug

Warning: Secure Debug is only supported on MPS3.

The MPS3 supports Authenticated Debug Access Control (ADAC), using the CoreSight SDC-600 IP.

For more information about this, see the following resources:

- [CoreSight SDC-600](#)
- [Authenticated Debug Access Control Specification](#)
- [Arm Corstone-1000 for MPS3 Application Note AN550, Chapter 7](#)

The Secure Debug Manager API is implemented in the [secure-debug-manager](#) repository. This repository also contains the necessary files for the Arm Development Studio support. The build and integration instructions can be found in its [README](#).

The *secure-debug-manager* repository also contains the private key and chain certificate to be used during the tests. The private key's public pair is provisioned into the One-Time Programmable memory in TrustedFirmware-M. These are dummy keys that should not be used in production.

To test the Secure Debug feature, you'll need a debug probe from the DSTREAM family and Arm Development Studio versions 2022.2, 2022.c, or 2023.a.

1. Clone the *secure-debug-manager* repository to your workspace.

```
cd $WORKSPACE
git clone https://github.com/ARM-software/secure-debug-manager.git
```

2. Navigate into the repository directory and checkout the specific commit in the listing below.

```
cd $WORKSPACE/secure-debug-manager
git checkout b30d6496ca749123e86b39b161b9f70ef76106d6
```

3. Follow the steps in the *secure-debug-manager*'s [README](#) for the development machine setup.
4. Rebuild the software stack with Secure Debug.

```
kas build meta-arm/kas/corstone1000-mps3.yml:meta-arm/ci/debug.yml:meta-arm/
↪ci/secure-debug.yml
```

5. Flash the firmware image as shown [here](#).
6. Run the software as shown [here](#).
7. Wait until the Secure Enclave terminal (ttyUSB1) prints the following prompts:

```
IComPortInit          : 382 : warn  : init      : IComPortInit:↵
↪Blocked reading of LPH2RA is active.
IComPortInit          : 383 : warn  : init      : IComPortInit:↵
↪Blocked reading LPH2RA
```

8. Connect the debug probe to the MPS3 using the 20-pin 1.27mm connector with the CS_20W_1.27MM silkscreen label.
9. Create a debug configuration in Arm Development Studio as described in the *secure-debug-manager*'s [README](#).
10. Connect the debugger to the target using the debug configuration.
11. Provide the paths to the private key and trust chain certificate when asked by Arm Development Studio Console.

```
...

Please provide private key file path:
Enter file path > $WORKSPACE\secure-debug-manager\example\data\keys\
↪EcdsaP256Key-3.pem

Please provide trust chain file path:
Enter file path > $WORKSPACE\secure-debug-manager\example\data\chains\chain.
↪EcdsaP256-3

...
```

12. When successful authenticated, Arm Development Studio will connect to the running MS3 and the debug features can be used. The following prompt should appear in the Secure Enclave terminal (ttyUSB1):

```
...
boot_platform_init: Corstone-1000 Secure Debug is a success.
...
```

1.3.10 Reports

Various test reports for the [Corstone-1000 software \(CORSTONE1000-2024.11\)](#) release version are available for reference [here](#).

Copyright (c) 2022-2024, Arm Limited. All rights reserved.

1.4 Release notes

1.4.1 Disclaimer

You expressly assume all liabilities and risks relating to your use or operation of Your Software and Your Hardware designed or modified using the Arm Tools, including without limitation, Your software or Your Hardware designed or intended for safety-critical applications. Should Your Software or Your Hardware prove defective, you assume the entire cost of all necessary servicing, repair or correction.

1.4.2 Release notes - 2024.11

The same notes as the 2024.06 release still apply.

1.4.3 Release notes - 2024.06

Known Issues or Limitations

- Use Ethernet over VirtIO due to lan91c111 Ethernet driver support dropped from U-Boot.
- Due to the performance uplimit of MPS3 FPGA and FVP, some Linux distros like Fedora Rawhide can not boot on Corstone-1000 (i.e. user may experience timeouts or boot hang).
- Corstone-1000 SoC on FVP doesn't have a secure debug peripheral. It does on the MPS3.
- See previous release notes for the known limitations regarding ACS tests.

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v2 <https://developer.arm.com/downloads/-/download-fpga-images>
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.23_25 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

1.4.4 Release notes - 2023.11

Known Issues or Limitations

- Use Ethernet over VirtIO due to lan91c111 Ethernet driver support dropped from U-Boot.
- Temporally removing the External system support in Linux due to it using multiple custom devicetree bindings that caused problems with SystemReady IR 2.0 certification. For External system support please refer to the version 2023.06. We are aiming to restore it in a more standardised manner in our next release.
- Due to the performance uplimit of MPS3 FPGA and FVP, some Linux distros like Fedora Rawhide can not boot on Corstone-1000 (i.e. user may experience timeouts or boot hang).
- PSA Crypto tests (psa-crypto-api-test command) approximately take 30 minutes to complete for FVP and MPS3.
- Corstone-1000 SoC on FVP doesn't have a secure debug peripheral. It does on the MPS3.
- See previous release notes for the known limitations regarding ACS tests.

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v2 <https://developer.arm.com/downloads/-/download-fpga-images>
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.23_25 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

1.4.5 Release notes - 2023.06

Known Issues or Limitations

- FPGA supports Linux distro install and boot through installer. However, FVP only supports openSUSE raw image installation and boot.
- Due to the performance uplimit of MPS3 FPGA and FVP, some Linux distros like Fedora Rawhide can not boot on Corstone-1000 (i.e. user may experience timeouts or boot hang).
- PSA Crypto tests (psa-crypto-api-test command) take 30 minutes to complete for FVP and 1 hour for MPS3.
- Corstone-1000 SoC on FVP doesn't have a secure debug peripheral. It does on the MPS3 .
- The following limitations listed in the previous release are still applicable:
 - UEFI Compliant - Boot from network protocols must be implemented – FAILURE
 - Known limitations regarding ACS tests - see previous release's notes.

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v2 <https://developer.arm.com/downloads/-/download-fpga-images>
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.19_21 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

1.4.6 Release notes - 2022.11.23

Known Issues or Limitations

- The external-system can not be reset individually on (or using) AN550_v1 FPGA release. However, the system-wide reset still applies to the external-system.
- FPGA supports Linux distro install and boot through installer. However, FVP only supports openSUSE raw image installation and boot.
- Due to the performance uplimit of MPS3 FPGA and FVP, some Linux distros like Fedora Rawhide can not boot on Corstone-1000 (i.e. user may experience timeouts or boot hang).
- Below SCT FAILURE is a known issues in the FVP: UEFI Compliant - Boot from network protocols must be implemented – FAILURE
- Below SCT FAILURE is a known issue when a terminal emulator (in the system where the user connects to serial ports) does not support 80x25 or 80x50 mode: EFI_SIMPLE_TEXT_OUT_PROTOCOL.SetMode - SetMode() with valid mode – FAILURE

- Known limitations regarding ACS tests: The behavior after running ACS tests on FVP is not consistent. Both behaviors are expected and are valid; The system might boot till the Linux prompt. Or, the system might wait after finishing the ACS tests. In both cases, the system executes the entire test suite and writes the results as stated in the user guide.

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v1 <https://developer.arm.com/downloads/-/download-fpga-images>
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.19_21 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

1.4.7 Release notes - 2022.04.04

Known Issues or Limitations

- FPGA support Linux distro install and boot through installer. However, FVP only support openSUSE raw image installation and boot.
- Due to the performance uplimit of MPS3 FPGA and FVP, some Linux distros like Fedora Rawhide cannot boot on Corstone-1000 (i.e. user may experience timeouts or boot hang).
- Below SCT FAILURE is a known issues in the FVP: UEFI Compliant - Boot from network protocols must be implemented – FAILURE

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v1
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.17_23 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

1.4.8 Release notes - 2022.02.25

Known Issues or Limitations

- The following tests only work on Corstone-1000 FPGA: ACS tests (SCT, FWTS, BSA), manual capsule update test, Linux distro install and boot.

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v1
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.17_23 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

Release notes - 2022.02.21

Known Issues or Limitations

- The following tests only work on Corstone-1000 FPGA: ACS tests (SCT, FWTS, BSA), manual capsule update test, Linux distro install and boot, psa-arch-test.

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v1
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.16.21 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

Release notes - 2022.01.18

Known Issues or Limitations

- Before running each SystemReady-IR tests: ACS tests (SCT, FWTS, BSA), manual capsule update test, Linux distro install and boot, etc., the SecureEnclave flash must be cleaned. See user-guide “Clean Secure Flash Before Testing” section.

Release notes - 2021.12.15

Software Features

The following components are present in the release:

- Yocto version Honister
- Linux kernel version 5.10
- U-Boot 2021.07
- OP-TEE version 3.14
- Trusted Firmware-A 2.5
- Trusted Firmware-M 1.5
- OpenAMP 347397decaa43372fc4d00f965640ebde042966d
- Trusted Services a365a04f937b9b76ebb2e0eeade226f208cbc0d2

Platform Support

- This software release is tested on Corstone-1000 FPGA version AN550_v1
- This software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.16.21 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

Known Issues or Limitations

- The following tests only work on Corstone-1000 FPGA: ACS tests (SCT, FWTS, BSA), manual capsule update test, Linux distro install and boot, and psa-arch-tests.
- Only the manual capsule update from UEFI shell is supported on FPGA.
- Due to flash size limitation and to support A/B banks, the wic image provided by the user should be smaller than 15MB.
- The failures in PSA Arch Crypto Test are known limitations with crypto library. It requires further investigation. The user can refer to [PSA Arch Crypto Test Failure Analysis In TF-M V1.5 Release](#) for the reason for each failing test.

Release notes - 2021.10.29

Software Features

This initial release of Corstone-1000 supports booting Linux on the Cortex-A35 and TF-M/MCUBOOT in the Secure Enclave. The following components are present in the release:

- Linux kernel version 5.10
- U-Boot 2021.07
- OP-TEE version 3.14
- Trusted Firmware-A 2.5
- Trusted Firmware-M 1.4

Platform Support

- This Software release is tested on Corstone-1000 Fast Model platform (FVP) version 11.16.21 <https://developer.arm.com/tools-and-software/open-source-software/arm-platforms-software/arm-ecosystem-fvps>

Known Issues or Limitations

- No software support for external system(Cortex M3)
- No communication established between A35 and M0+
- Very basic functionality of booting Secure Enclave, Trusted Firmware-A , OP-TEE , u-boot and Linux are performed

Support

For technical support email: support-subsystem-iot@arm.com

For all security issues, contact Arm by email at psirt@arm.com.

Copyright (c) 2022-2023, Arm Limited. All rights reserved.

1.5 Change Log

This document contains a summary of the new features, changes and fixes in each release of Corstone-1000 software stack.

1.5.1 Version 2024.11

Changes

- Implementation of a replication strategy for FWU metadata in TF-M according to the FWU specification.
- Upgrade to metadata version 2 in TF-M.
- Increase the ITS and PS memory size in Secure Flash for TF-M.
- SW components upgrades.
- Bug fixes.

Corstone-1000 components versions

linux-yocto	6.10.14
u-boot	2023.07.02
external-system	0.1.0
optee-client	4.2.0
optee-os	4.2.0
trusted-firmware-a	2.11.0
trusted-firmware-m	2.1.0
libts	602be60719
ts-newlib	4.1.0
ts-psa-{crypto, iat, its, ps}-api-test	74dc6646ff
ts-sp-{se-proxy, smm-gateway}	602be60719

Yocto distribution components versions

meta-arm	styhead
poky	5465094be9
meta-openembedded	461d85a183
meta-secure-core	59d7e90542
busybox	1.36.1
musl	1.2.5
gcc-arm-none-eabi	13.3.re11
gcc-cross-aarch64	14.2.0
openssl	3.3.1

1.5.2 Version 2024.06

Changes

- Re-enabling support for the External System using linux remoteproc (only supporting switching on and off the External System)
- UEFI Secure Boot and Authenticated Variable support
- RSE Comms replaces OpenAMP
- The EFI System partition image is now created by the meta-arm build system. This image is mounted on the second MMC card by default in the FVP.
- The capsule generation script is now part of the meta-arm build system. Corstone1000-flash-firmware-image recipe generates a capsule binary using the U-Boot capsule generation tool that includes all the firmware binaries and recovery kernel image.
- SW components upgrades
- Bug fixes

Corstone-1000 components versions

arm-tstee	2.0.0
linux-yocto	6.6.23
u-boot	2023.07.02
external-system	0.1.0
optee-client	4.1.0
optee-os	4.1.0
trusted-firmware-a	2.10.4
trusted-firmware-m	2.0.0
libts	602be60719
ts-newlib	4.1.0
ts-psa-{crypto, iat, its, ps}-api-test	602be60719
ts-sp-{se-proxy, smm-gateway}	602be60719

Yocto distribution components versions

meta-arm	scarthgap
poky	scarthgap
meta-openembedded	scarthgap
meta-secure-core	scarthgap
busybox	1.36.1
musl	1.2.4
gcc-arm-none-eabi	13.2.Rel1
gcc-cross-aarch64	13.2.0
openssl	3.2.1

1.5.3 Version 2023.11

Changes

- Making Corstone-1000 SystemReady IR 2.0 certifiable
- Allow booting Debian & OpenSUSE on FVP
- Add support for two MMC cards for the FVP
- Add signed capsule update support
- Enable on-disk capsule update
- Add the feature of purging specific DT nodes in U-Boot before Linux
- Add Ethernet over VirtIO support in U-Boot
- Add support for unaligned MMC card images
- Reducing the out-of-tree patches by upstreaming them to the corresponding open-source projects
- SW components upgrades
- Bug fixes

Corstone-1000 components versions

arm-ffa-tee	1.1.2-r0
linux-yocto	6.5.7
u-boot	2023.07
external-system	0.1.0+gitAUTOINC+8c9dca74b1-r0
optee-client	3.22.0
optee-os	3.22.0
trusted-firmware-a	2.9.0
trusted-firmware-m	1.8.1
libts	08b3d39471
ts-newlib	4.1.0
ts-psa-{crypto, iat, its, ps}-api-test	38cb53a4d9
ts-sp-{se-proxy, smm-gateway}	08b3d39471

Yocto distribution components versions

meta-arm	nanbiel
poky	nanbiel
meta-openembedded	nanbiel
meta-secure-core	nanbiel
busybox	1.36.1
musl	1.2.4
gcc-arm-none-eabi	11.2-2022.02
gcc-cross-aarch64	13.2.0
openssl	3.1.3

1.5.4 Version 2023.06

Changes

- GPT support (in TF-M, TF-A, U-boot)
- Use TF-M BL1 code as the ROM code instead of MCUboot (the next stage bootloader BL2 remains to be MCU-boot)
- Secure Enclave uses CC312 OTP as the provisioning backend in FVP and FPGA
- NVMMXIP block storage support in U-Boot
- Upgrading the SW stack recipes
- Upgrades for the U-Boot FF-A driver and MM communication

Corstone-1000 components versions

arm-ffa-tee	1.1.2-r0
arm-ffa-user	5.0.1-r0
corstone1000-external-sys-tests	1.0+gitAUTOINC+2945cd92f7-r0
external-system	0.1.0+gitAUTOINC+8c9dca74b1-r0
linux-yocto	6.1.25+gitAUTOINC+36901b5b29_581dc1aa2f-r0
u-boot	2023.01-r0
optee-client	3.18.0-r0
optee-os	3.20.0-r0
trusted-firmware-a	2.8.0-r0
trusted-firmware-m	1.7.0-r0
ts-newlib	4.1.0-r0
ts-psa-{crypto, iat, its, ps}-api-test	38cb53a4d9
ts-sp-{se-proxy, smm-gateway}	08b3d39471

Yocto distribution components versions

meta-arm	mickledore
poky	mickledore
meta-openembedded	mickledore
busybox	1.36.0-r0
musl	1.2.3+gitAUTOINC+7d756e1c04-r0
gcc-arm-none-eabi-native	11.2-2022.02
gcc-cross-aarch64	12.2.rel1-r0
openssl	3.1.0-r0

1.5.5 Version 2022.11.23

Changes

- Booting the External System (Cortex-M3) with RTX RTOS
- Adding MHU communication between the HOST (Cortex-A35) and the External System
- Adding a Linux application to test the External System
- Adding ESRT (EFI System Resource Table) support
- Upgrading the SW stack recipes
- Upgrades for the U-Boot FF-A driver and MM communication

Corstone-1000 components versions

arm-ffa-tee	1.1.1
arm-ffa-user	5.0.0
corstone1000-external-sys-tests	1.0
external-system	0.1.0
linux-yocto	5.19
u-boot	2022.07
optee-client	3.18.0
optee-os	3.18.0
trusted-firmware-a	2.7.0
trusted-firmware-m	1.6.0
ts-newlib	4.1.0
ts-psa-{crypto, iat, its, ps}-api-test	451aa087a4
ts-sp-{se-proxy, smm-gateway}	3d4956770f

Yocto distribution components versions

meta-arm	langdale
poky	langdale
meta-openembedded	langdale
busybox	1.35.0
musl	1.2.3+git37e18b7bf3
gcc-arm-none-eabi-native	11.2-2022.02
gcc-cross-aarch64	12.2
openssl	3.0.5

1.5.6 Version 2022.04.04

Changes

- Linux distro openSUSE, raw image installation and boot in the FVP.
- SCT test support in FVP.
- Manual capsule update support in FVP.

1.5.7 Version 2022.02.25

Changes

- Building and running psa-arch-tests on Corstone-1000 FVP
- Enabled smm-gateway partition in Trusted Service on Corstone-1000 FVP
- Enabled MHU driver in Trusted Service on Corstone-1000 FVP
- Enabled OpenAMP support in SE proxy SP on Corstone-1000 FVP

1.5.8 Version 2022.02.21

Changes

- psa-arch-tests: recipe is dropped and merged into the secure-partitons recipe.
- psa-arch-tests: The tests are align with latest tfm version for psa-crypto-api suite.

1.5.9 Version 2022.01.18

Changes

- psa-arch-tests: change master to main for psa-arch-tests
- U-Boot: fix null pointer exception for get_image_info
- TF-M: fix capsule instability issue for Corstone-1000

1.5.10 Version 2022.01.07

Changes

- Corstone-1000: fix SystemReady-IR ACS test (SCT, FWTS) failures.
- U-Boot: send bootcomplete event to secure enclave.
- U-Boot: support populating Corstone-1000 image_info to ESRT table.
- U-Boot: add ethernet device and enable configs to support bootfromnetwork SCT.

1.5.11 Version 2021.12.15

Changes

- Enabling Corstone-1000 FPGA support on: - Linux 5.10 - OP-TEE 3.14 - Trusted Firmware-A 2.5 - Trusted Firmware-M 1.5
- Building and running psa-arch-tests
- Adding openamp support in SE proxy SP
- OP-TEE: adding smm-gateway partition
- U-Boot: introducing Arm FF-A and MM support

1.5.12 Version 2021.10.29

Changes

- Enabling Corstone-1000 FVP support on: - Linux 5.10 - OP-TEE 3.14 - Trusted Firmware-A 2.5 - Trusted Firmware-M 1.4
 - Linux kernel: enabling EFI, adding FF-A debugfs driver, integrating ARM_FFA_TRANSPORT.
 - U-Boot: Extending EFI support
 - python3-imgtool: adding recipe for Trusted-firmware-m
 - python3-imgtool: adding the Yocto recipe used in signing host images (based on MCUBOOT format)
-

Copyright (c) 2022-2024, Arm Limited. All rights reserved.