corstone1000

Arm Limited

May 03, 2024

CONTENTS

CHAPTER

ONE

SOFTWARE ARCHITECTURE

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.

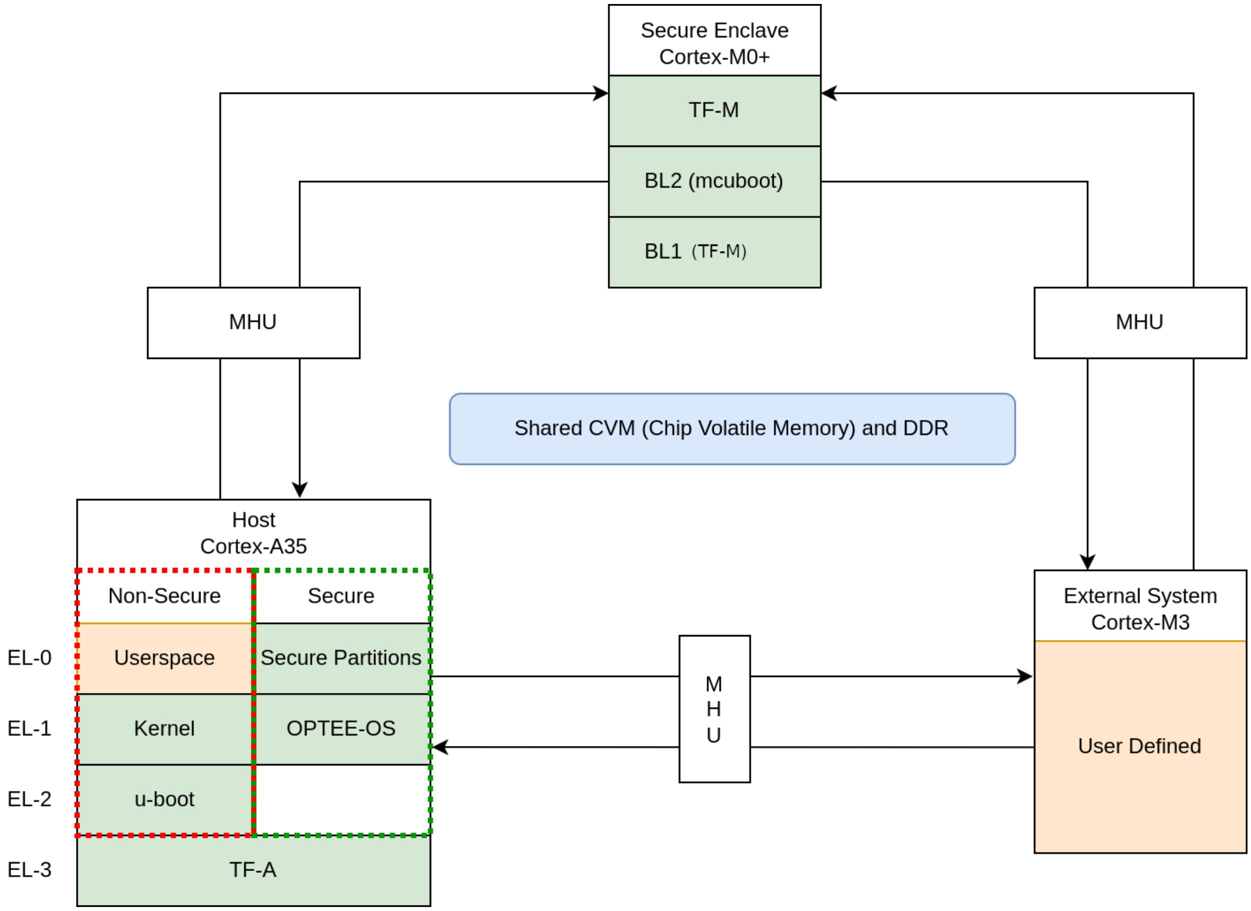
Corstone-1000 software plus hardware reference solution is PSA Level-2 ready certified (PSA L2 Ready) as well as System Ready IR certified(SRIR cert). More information on the Corstone-1000 subsystem product and design can be found at: Arm corstone1000 Software and Arm corstone1000 Technical Overview.

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.

1.2 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 Encalve 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.

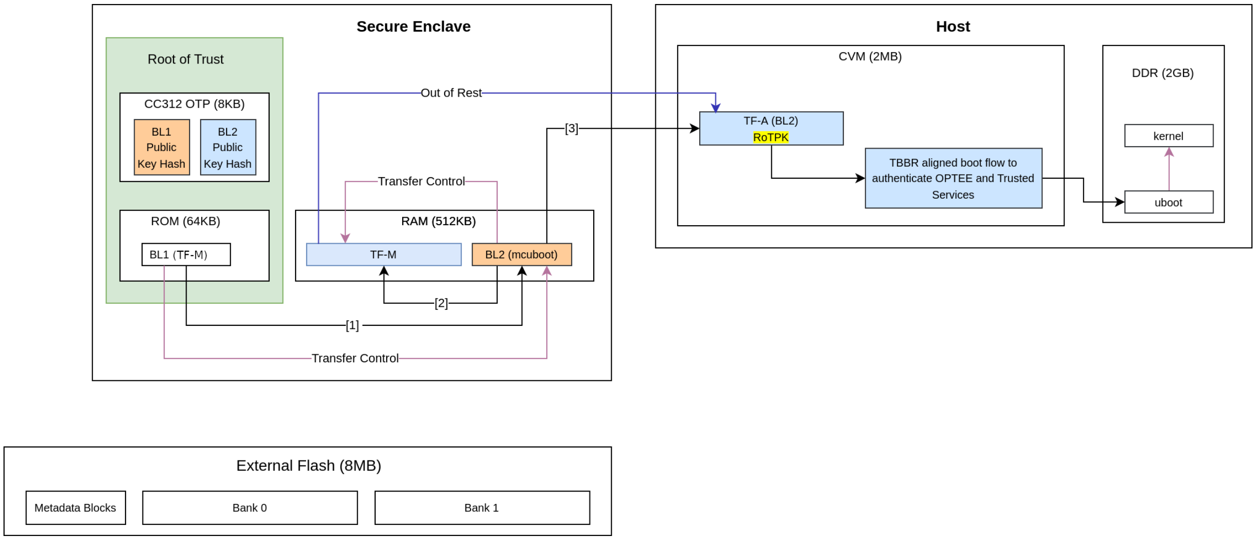
The Host System is based on ARM Cotex-A35 processor with standardized peripherals to allow for the booting of a Linux OS. The Cortex-A35 has the TrustZone technology that allows secure and non-secure security states in the processor. The software design in the Host System follows Firmware Framework for A class procseeor (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) is performed using MHU as transport mechanism and rpmsg messaging system (the external system support in Linux is disabled in this release. More info about this change can be found in the release-notes).

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 connectivitiy, and edge computing.

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 BL1 image of the secure enclave and content of the CC312 OTP (One Time Programmable) memory forms the RoT. The BL1 image exists in ROM (Read Only Memory).



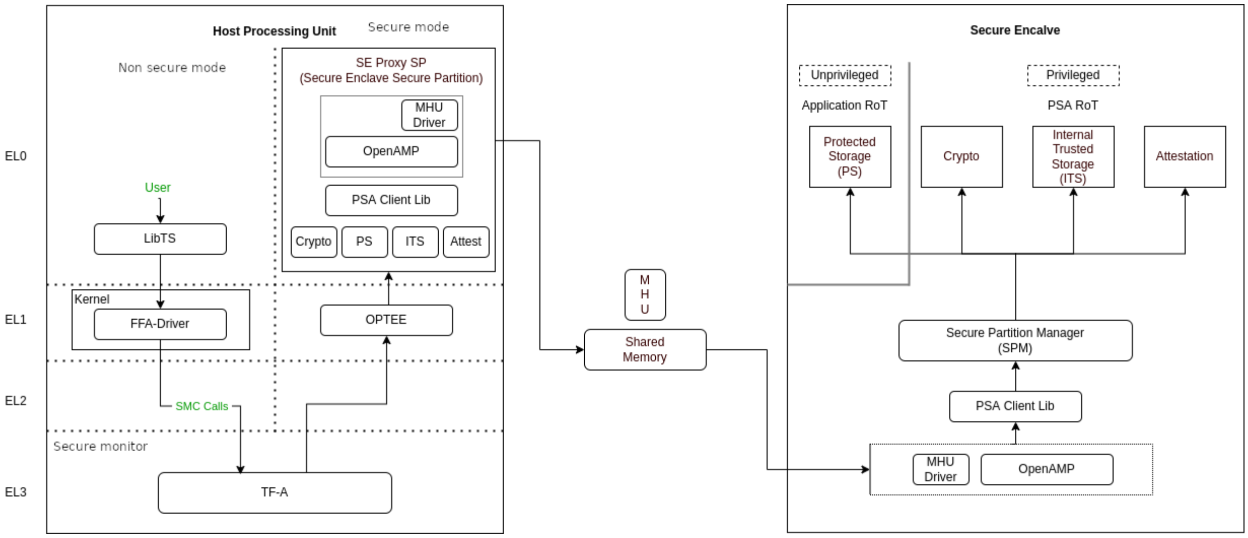
It is a lengthy chain to boot the software on Corstone-1000. On power on, the secure enclave starts executing BL1 code from the ROM which is the RoT of the device. Authentication of an image involves the steps listed below:

- Load image from flash 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.

In the secure enclave, BL1 authenticates the BL2 and passes the execution control. BL2 authenticates the initial boot loader of the host (Host TF-A BL2) and TF-M. The execution control is now passed to TF-M. TF-M being the run time executable of secure enclave which initializes itself and, at the end, brings the host CPU out of rest. The host follows the boot standard defined in the TBBR to authenticate the secure and non-secure software.

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.



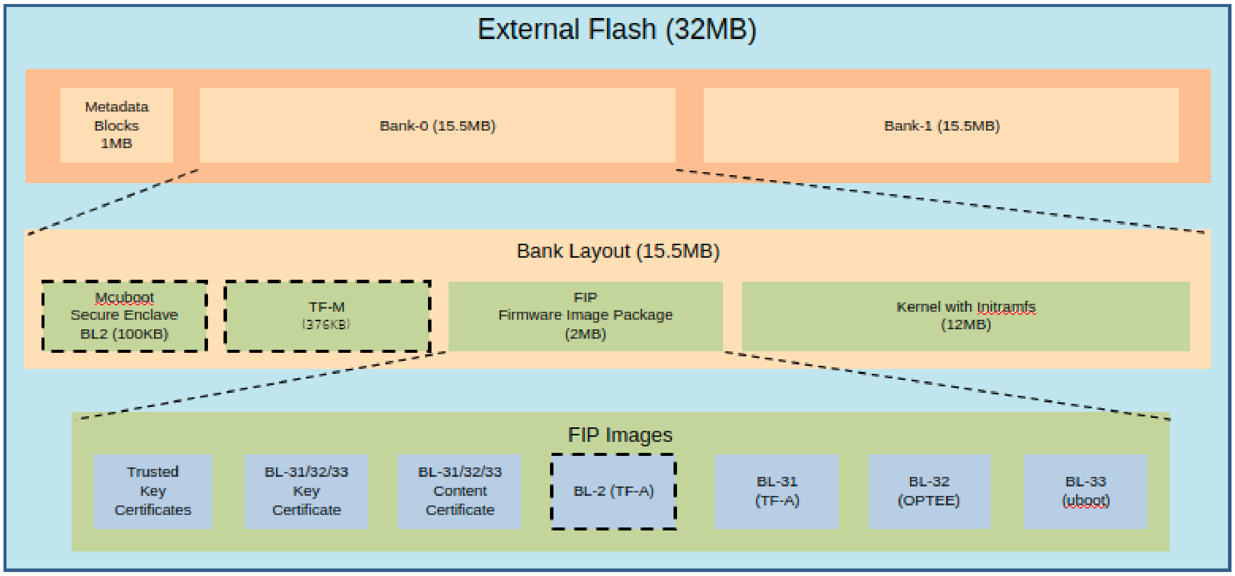
The SE Proxy SP (Secure Enclave Proxy Secure Partition) is a proxy partition managed by OPTEE which forwards such calls to the secure enclave. The solution relies on OpenAMP which uses shared memory and MHU interrupts as a doorbell for communication between two cores. Corstone-1000 implements isolation 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 encalve 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.

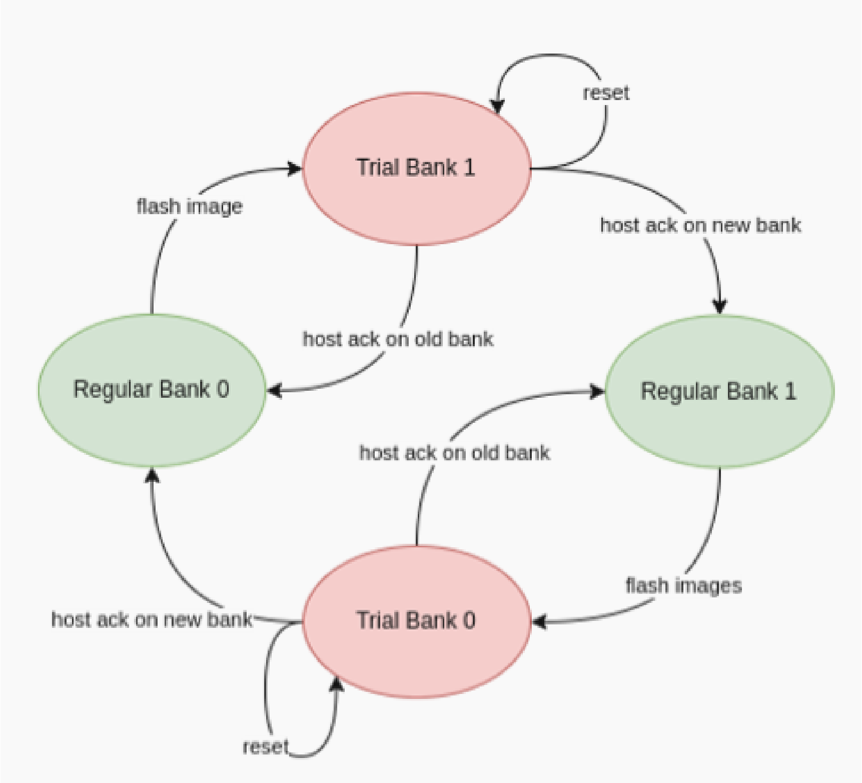
1.5 Secure Firmware Update

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.

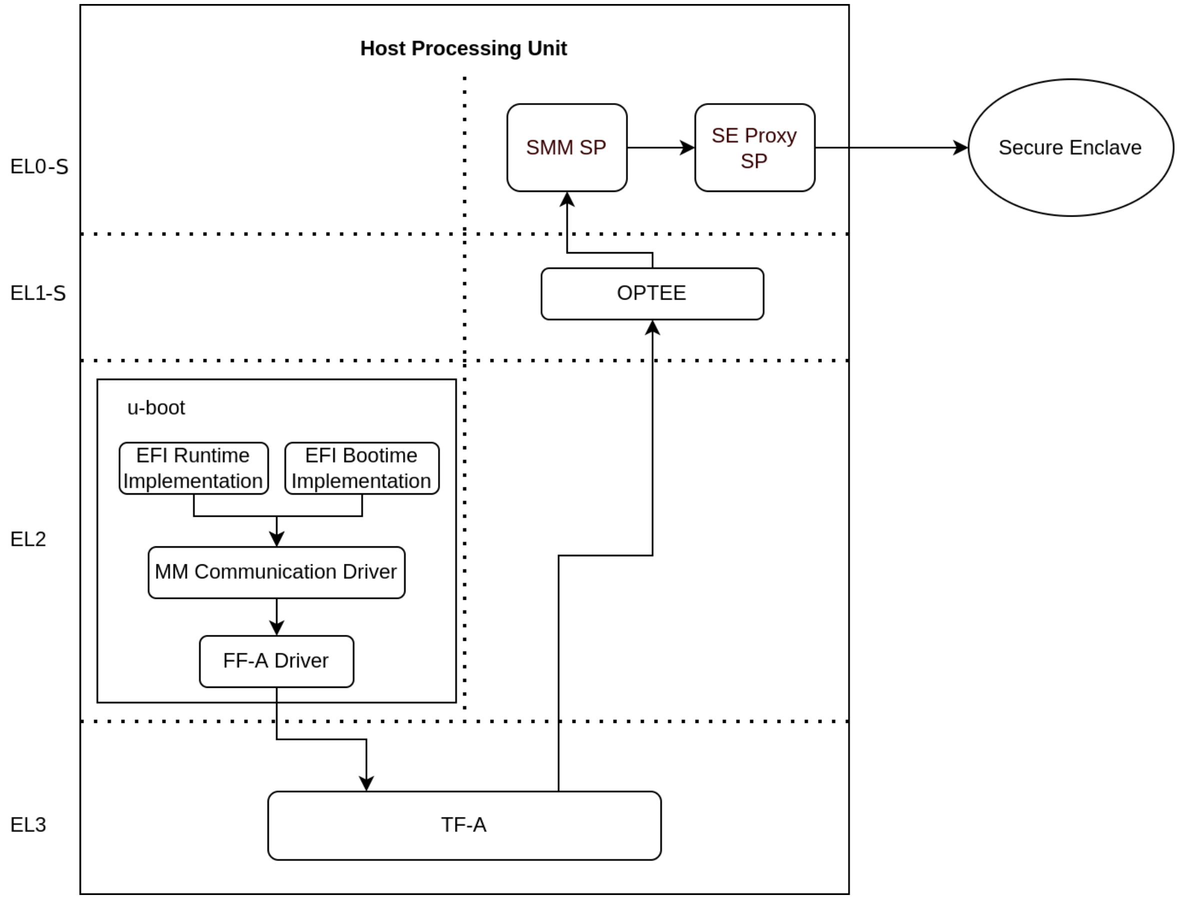


When Firmware update is triggered, u-boot verifies the capsule by checking the capsule signature, version number and size. Then it signals the Secure Enclave that can start writing UEFI capsule into the flash. Once this operation finishes ,Secure Enclave resets the entire system. The Metadata Block in 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.



1.6 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.



1.7 References

ARM corstone1000 Search

Arm security features

CHAPTER

TWO

USER GUIDE

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 initramfs 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.

2.2 Prerequisites

This guide assumes that your host machine is running Ubuntu 20.04 LTS, 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
- tar 1.28 or greater
- Python 3.8.0 or greater.
- gcc 8.0 or greater.
- GNU make 4.0 or greater

Please follow the steps described in the Yocto mega manual:

- Compatible Linux Distribution
- Build Host Packages

2.3 Targets

- Arm Corstone-1000 Ecosystem FVP (Fixed Virtual Platform)
- Arm Corstone-1000 for MPS3

2.4 Yocto stable branch

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

2.5 Provided 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/`.

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

- `<_workspace>/meta-arm/meta-arm-bsp/conf/machine/include/corstone1000.inc`
- `<_workspace>/meta-arm/meta-arm-bsp/conf/machine/corstone1000-fvp.conf`
- `<_workspace>/meta-arm/meta-arm-bsp/conf/machine/corstone1000-mps3.conf`

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

2.5.1 Software for Host

Trusted Firmware-A

Based on 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_2.9.0.bb

OP-TEE

Based on OP-TEE

bbappend	<_workspace>/meta-arm/meta-arm-bsp/recipes-security/optee/optee-os_3.22.0.bbappend
Recipe	<_workspace>/meta-arm/meta-arm-bsp/recipes-security/optee/optee-os_3.22.0.bb

U-Boot

Based on U-Boot repo

bbappend	<_workspace>/meta-arm/meta-arm/recipes-bsp/u-boot/u-boot_%.bbappend
bbappend	<_workspace>/meta-arm/meta-arm-bsp/recipes-bsp/u-boot/u-boot_%.bbappend
Recipe	<_workspace>/meta-arm/meta-arm-bsp/recipes-bsp/u-boot/u-boot_2023.07.02.bb

Linux

The distro is based on the poky-tiny distribution which is a Linux distribution stripped down to a minimal configuration.

The provided distribution is based on busybox and built using musl libc. The recipe responsible for building a tiny version of Linux is listed below.

bbappend	<_workspace>/meta-arm/meta-arm-bsp/recipes-kernel/linux/linux-yocto_%.bbappend
Recipe	<_workspace>/poky/meta/recipes-kernel/linux/linux-yocto_6.5.bb
defconfig	<_workspace>/meta-arm/meta-arm-bsp/recipes-kernel/linux/files/corstone1000/defconfig

2.5.2 Software for Boot Processor (a.k.a Secure Enclave)

Based on Trusted Firmware-M

bbappend	<_workspace>/meta-arm/meta-arm-bsp/recipes-bsp/trusted-firmware-m/trusted-firmware-m_%.bbappend
Recipe	<_workspace>/meta-arm/meta-arm/recipes-bsp/trusted-firmware-m/trusted-firmware-m_1.8.1.bb

2.5.3 Software for the External System

RTX

Based on RTX RTOS

Recipe	<_workspace>/meta-arm/meta-arm-bsp/recipes-bsp/external-system/external-system_0.1.0.bb

2.6 Building the software stack

Create a new folder that will be your workspace and will henceforth be referred to as <_workspace> in these instructions. To create the folder, run:

```
mkdir <_workspace>
cd <_workspace>
```

Corstone-1000 software is based on the Yocto Project which uses kas and bitbake commands to build the stack. kas version 4 is required. To install kas, run:

```
pip3 install kas
```

If 'kas' command is not found in command-line, please make sure the user installation directories are visible on $PATH. If you have sudo rights, try 'sudo pip3 install kas'.

In the top directory of the workspace <_workspace>, run:

```
git clone https://git.yoctoproject.org/git/meta-arm -b CORSTONE1000-2023.11
```

To build a Corstone-1000 image for MPS3 FPGA, run:

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

Alternatively, to build a Corstone-1000 image for FVP, you need to accept the EULA at https://developer.arm.com/downloads/-/arm-ecosystem-fvps/eula by setting the ARM_FVP_EULA_ACCEPT environment variable as follows:

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

then run:

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

The initial clean build will be lengthy, given that all host utilities are to be built as well as the target images. This includes host executables (python, cmake, etc.) and the required toolchain(s).

Once the build is successful, all output binaries will be placed in the following folders:

- `<_workspace>/build/tmp/deploy/images/corstone1000-fvp/` folder for FVP build;
- `<_workspace>/build/tmp/deploy/images/corstone1000-mps3/` folder for FPGA build.

Everything apart from the Secure Enclave ROM firmware and External System firmware, is bundled into a single binary, the `corstone1000-image-corstone1000-{mps3,fvp}.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-{mps3,fvp}/bl1.bin`
- The External System firmware: `<_workspace>/build/tmp/deploy/images/corstone1000-{mps3,fvp}/es_flashfw.bin`
- The flash image: `<_workspace>/build/tmp/deploy/images/corstone1000-{mps3,fvp}/corstone1000-image-corstone1000-{mps3,fvp}.wic`

2.7 Flash the firmware image on FPGA

The user should download the FPGA bit file image `AN550:  Arm® Corstone™-1000 for MPS3 Version 2.0` from this link and under the section `Arm® Corstone™-1000 for MPS3`. The download is available after logging in.

The directory structure of the FPGA bundle is shown below.

```
Boardfiles
├── config.txt
├── MB
│   ├── BRD_LOG.TXT
│   ├── HBI0309B
│   │   ├── AN550
│   │   │   ├── AN550_v2.bit
│   │   │   ├── an550_v2.txt
│   │   │   └── images.txt
│   │   ├── board.txt
│   │   └── mbb_v210.ebf
│   └── HBI0309C
│       ├── AN550
│       │   ├── AN550_v2.bit
```

(continues on next page)

(continued from previous page)

```
│       │   ├── an550_v2.txt
│       │   └── images.txt
│       ├── board.txt
│       └── mbb_v210.ebf
└── SOFTWARE
    ├── an550_st.axf
    ├── bl1.bin
    ├── cs1000.bin
    └── ES0.bin
```

Depending upon the MPS3 board version (printed on the MPS3 board) you should update the images.txt file (in corresponding HBI0309x folder. Boardfiles/MB/HBI0309<board_revision>/AN550/images.txt) so that the file points to the images under SOFTWARE directory.

The images.txt file that is 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 OCVM (DDR4 2GB)
;  PORT 1        Secure Enclave (M0+) ROM (64KB)
;  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\bl1.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
```

OUTPUT_DIR = `<_workspace>/build/tmp/deploy/images/corstone1000-mps3`

1. Copy `bl1.bin` from OUTPUT_DIR directory to SOFTWARE directory of the FPGA bundle.
2. Copy `es_flashfw.bin` from OUTPUT_DIR directory to SOFTWARE directory of the FPGA bundle and rename the binary to `es0.bin`.
3. Copy `corstone1000-image-corstone1000-mps3.wic` from OUTPUT_DIR directory to SOFTWARE directory of the FPGA bundle and rename the wic image to `cs1000.bin`.

NOTE: Renaming of the images are required because MCC firmware has limitation of 8 characters before .(dot) and 3 characters after .(dot).

Now, copy the entire folder to board's SDCard and reboot the board.

2.8 Running the software on FPGA

On the host machine, open 4 serial port terminals. In case of Linux machine it will be ttyUSB0, ttyUSB1, ttyUSB2, ttyUSB3 and it might be different on Windows machines.

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

Run following commands to open serial port terminals on Linux:

```
sudo picocom -b 115200 /dev/ttyUSB0  # in one terminal
sudo picocom -b 115200 /dev/ttyUSB1  # in another terminal
sudo picocom -b 115200 /dev/ttyUSB2  # in another terminal.
sudo picocom -b 115200 /dev/ttyUSB3  # in another terminal.
```

NOTE: The MPS3 expects an ethernet cable to be plugged in, otherwise it will wait for the network for a considerable amount of time, printing the following logs:

```
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 0xffffffc008e50000, IRQ: 17
Waiting up to 100 more seconds for network.
```

Once the system boot is completed, you should see console logs on the serial port terminals. Once the HOST(Cortex-A35) is booted completely, user can login to the shell using **"root"** login.

If system does not boot and only the ttyUSB1 logs are visible, please follow the steps in *Clean Secure Flash Before Testing (applicable to FPGA only)* under *SystemReady-IR tests* section. The previous image used in FPGA (MPS3) might have filled the Secure Flash completely. The best practice is to clean the secure flash in this case.

2.9 Running the software on FVP

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

A Yocto recipe is provided and allows 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 supported Fixed Virtual Platform (FVP) version is 11_23.25 and is automatically downloaded and installed when using the runfvp command as detailed below. The FVP version can be checked by running the following command:

```
<_workspace>/meta-arm/scripts/runfvp <_workspace>/build/tmp/deploy/images/corstone1000-
↪fvp/corstone1000-image-corstone1000-fvp.fvpconf -- --version
```

The FVP can also be manually downloaded from the Arm Ecosystem FVPs page. On this page, navigate to "Corstone IoT FVPs" section to download the Corstone-1000 platform FVP installer. Follow the instructions of the installer and setup the FVP.

To run the FVP using the runfvp command, please run the following command:

```
<_workspace>/meta-arm/scripts/runfvp --terminals=xterm <_workspace>/build/tmp/deploy/
↪images/corstone1000-fvp/corstone1000-image-corstone1000-fvp.fvpconf
```

When the script is executed, three terminal instances will be launched, one for the boot processor (aka Secure Enclave) processing element and two for the Host processing element. Once the FVP is executing, the Boot Processor will start to boot, wherein the relevant memory contents of the .wic file are copied to their respective memory locations within the model, enforce firewall policies on memories and peripherals and then, bring the host out of reset.

The host will boot trusted-firmware-a, OP-TEE, U-Boot and then Linux, and present a login prompt (FVP host_terminal_0):

```
corstone1000-fvp login:
```

Login using the username root.

2.10 SystemReady-IR tests

2.10.1 Testing steps

NOTE: Running the SystemReady-IR tests described below requires the user to work with USB sticks. In our testing, not all USB stick models work well with MPS3 FPGA. Here are the USB sticks models that are 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

NOTE: Before running each of the tests in this chapter, the user should follow the steps described in following section "Clean Secure Flash Before Testing" to erase the SecureEnclave flash cleanly and prepare a clean board environment for the testing.

Prepare EFI System Partition

Corstone-1000 FVP and FPGA do not have enough on-chip nonvolatile memory to host an EFI System Partition (ESP). Thus, Corstone-1000 uses mass storage device for ESP. The instructions below should be followed for both FVP and FPGA before running the ACS tests.

Common to FVP and FPGA:

1. Create an empty 100 MB partition:

   ```
   dd if=/dev/zero of=corstone1000-efi-partition.img iflag=fullblock bs=512␣
   ↪count=204800 && sync
   ```

2. Use OpenSuse Raw image to copy the contents of EFI partition.

   To download OpenSUSE Tumbleweed raw image:

   - Under OpenSUSE Tumbleweed appliances

- The user should look for a Tumbleweed-ARM-JeOS-efi.aarch64-* Snapshot, for example, `openSUSE-Tumbleweed-ARM-JeOS-efi.aarch64-<date>-Snapshot<date>.raw.xz`

Once the .raw.xz file is downloaded, the raw image file needs to be extracted:

```
unxz <file-name.raw.xz>
```

The above command will generate a file ending with extension .raw image. Use the following command to get address of the first partition

```
fdisk -lu <path-to-img>/openSUSE-Tumbleweed-ARM-JeOS-efi.aarch64-<date>-Snapshot
↪<date>.raw
-> Device                                                                   ␣
↪     Start     End  Sectors  Size Type
     <path-to-img>/openSUSE-Tumbleweed-ARM-JeOS-efi.aarch64-<date>-Snapshot<date>.
↪raw1     8192    40959    32768    16M EFI System
     <path-to-img>/openSUSE-Tumbleweed-ARM-JeOS-efi.aarch64-<date>-Snapshot<date>.
↪raw2    40960 1064959  1024000   500M Linux swap
     <path-to-img>/openSUSE-Tumbleweed-ARM-JeOS-efi.aarch64-<date>-Snapshot<date>.
↪raw3 1064960 5369822  4304863   2.1G Linux filesystem

->   <blockaddress_1st_partition> = 8192
->   <sectorsize_1st_partition> = 32768
```

3. Copy the ESP from opensuse image to empty image:

```
dd conv=notrunc if=openSUSE-Tumbleweed-ARM-JeOS-efi.aarch64-<date>-Snapshot<date>.
↪raw skip=<blockaddress_1st_partition> of=corstone1000-efi-partition.img seek=
↪<blockaddress_1st_partition> iflag=fullblock seek=<blockaddress_1st_partition>␣
↪bs=512 count=<sectorsize_1s_partition> && sync
```

4. Create the file efi_disk.layout locally. Copy the content of provided disk layout below to the efi_disk.layout to label the ESP correctly.

efi_disk.layout

```
label: gpt
label-id: AC53D121-B818-4515-9031-BE02CCEB8701
device: corstone1000-efi-partition.img
unit: sectors
first-lba: 34
last-lba: 204766

corstone1000-efi-partition.img : start=8192, size=32768, type=C12A7328-F81F-11D2-
↪BA4B-00A0C93EC93B, uuid=792D821F-98AE-46E3-BABD-948003A650F8, name="p.UEFI"
```

And use the following command the label the newly created ESP.

```
sfdisk corstone1000-efi-partition.img < efi_disk.layout
```

To test the image, you can now mount the disk image

```
fdisk -lu corstone1000-efi-partition.img
-> Device                          Start   End Sectors Size Type
   corstone1000-efi-partition.img1  8192 40959   32768  16M EFI System
```

(continues on next page)

(continued from previous page)

```
    <offset_1st_partition> = 8192 * 512 (sector size) = 4194304

sudo mount -o loop,offset=4194304 corstone1000-efi-partition.img /mount_point
```

Using ESP in FPGA:

Once the ESP is created, it needs to be flashed to a second USB drive different than ACS image. This can be done with the development machine. In the given example here we assume the USB device is `/dev/sdb` (the user should use `lsblk` command to confirm). Be cautious here and don't confuse your host machine own hard drive with the USB drive. Run the following commands to prepare the ACS image in USB stick:

```
sudo dd if=corstone1000-efi-partition.img of=/dev/sdb iflag=direct oflag=direct␣
↪status=progress bs=512; sync;
```

Now you can plug this USB stick to the board together with ACS test USB stick.

Using ESP in FVP:

The ESP disk image can directly be used in Corstone-1000 FVP by simply passing it as the 2nd MMC card image.

```
<_workspace>/meta-arm/scripts/runfvp <_workspace>/build/tmp/deploy/images/corstone1000-
↪fvp/corstone1000-image-corstone1000-fvp.fvpconf -- -C board.msd_mmc.p_mmc_file="${
↪<path-to-img>/ir_acs_live_image.img}" -C board.msd_mmc_2.p_mmc_file="${<path-to-img>/
↪corstone1000-efi-partition.img}"
```

Clean Secure Flash Before Testing (applicable to FPGA only)

To prepare a clean board environment with clean secure flash for the testing, the user should prepare an image that erases the secure flash cleanly during boot. Run following commands to build such image.

```
cd <_workspace>
git clone https://git.yoctoproject.org/git/meta-arm -b CORSTONE1000-2023.11
git clone https://git.gitlab.arm.com/arm-reference-solutions/systemready-patch.git -b␣
↪CORSTONE1000-2023.11
cp -f systemready-patch/embedded-a/corstone1000/erase_flash/0001-embedded-a-corstone1000-
↪clean-secure-flash.patch meta-arm
cd meta-arm
git apply 0001-embedded-a-corstone1000-clean-secure-flash.patch
cd ..
kas build meta-arm/kas/corstone1000-mps3.yml:meta-arm/ci/debug.yml
```

Replace the bl1.bin and cs1000.bin files on the SD card with following files:

- The ROM firmware: <_workspace>/build/tmp/deploy/images/corstone1000-mps3/bl1.bin
- The flash image: <_workspace>/build/tmp/deploy/images/corstone1000-mps3/corstone1000-image-corstone1000-mps3.wic

Now reboot the board. This step erases the Corstone-1000 SecureEnclave flash completely, the user should expect following message from TF-M log (can be seen in 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
```

Then the user should follow "Building the software stack" to build a clean software stack and flash the FPGA as normal. And continue the testing.

Run SystemReady-IR ACS tests

Architecture Compliance Suite (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 (For example: SBSA, SBBR, etc.), so that implementers can verify if these behaviours have been interpreted correctly.

ACS image contains two partitions. BOOT partition and RESULT partition. Following test suites and bootable applications are under BOOT partition:

- SCT
- FWTS
- BSA uefi
- BSA linux
- grub
- uefi manual capsule application

BOOT partition contains the following:

```
├── EFI
│   └── BOOT
│       ├── app
│       ├── bbr
│       ├── bootaa64.efi
│       ├── bsa
│       ├── debug
│       ├── Shell.efi
│       └── startup.nsh
├── grub
├── grub.cfg
├── Image
└── ramdisk-busybox.img
```

RESULT partition is used to store the test results. **NOTE**: PLEASE MAKE SURE THAT "acs_results" FOLDER UNDER THE RESULT PARTITION IS EMPTY BEFORE YOU START THE TESTING. OTHERWISE THE TEST RESULTS WILL NOT BE CONSISTENT

FPGA instructions for ACS image

This section describes how the user can build and run Architecture Compliance Suite (ACS) tests on Corstone-1000.

First, the user should download the Arm SystemReady ACS repository. This repository contains the infrastructure to build the Architecture Compliance Suite (ACS) and the bootable prebuilt images to be used for the certifications of SystemReady-IR. To download the repository, run command:

```
cd <_workspace>
git clone https://github.com/ARM-software/arm-systemready.git
```

Once the repository is successfully downloaded, the prebuilt ACS live image can be found in:

- `<_workspace>/arm-systemready/IR/prebuilt_images/v23.03_2.0.0/ir-acs-live-image-generic-arm64.wic.xz`

NOTE: This prebuilt ACS image includes v5.13 kernel, which doesn't provide USB driver support for Corstone-1000. The ACS image with newer kernel version and with full USB support for Corstone-1000 will be available in the next SystemReady release in this repository.

Then, the user should prepare a USB stick with ACS image. In the given example here, we assume the USB device is `/dev/sdb` (the user should use `lsblk` command to confirm). Be cautious here and don't confuse your host machine own hard drive with the USB drive. Run the following commands to prepare the ACS image in USB stick:

```
cd <_workspace>/arm-systemready/IR/prebuilt_images/v23.03_2.0.0
unxz ir-acs-live-image-generic-arm64.wic.xz
sudo dd if=ir-acs-live-image-generic-arm64.wic of=/dev/sdb iflag=direct oflag=direct␣
↪bs=1M status=progress; sync
```

Once the USB stick with ACS image is prepared, the user should make sure that ensure that both USB sticks (ESP and ACS image) are connected to the board, and then boot the board.

The FPGA will reset multiple times during the test, and it might take approx. 24-36 hours to finish the test.

NOTE: The USB stick which contains the ESP partition might cause grub to unable to find the bootable partition (only in the FPGA). If that's the case, please remove the USB stick and run the ACS tests. ESP partition can be mounted after the platform is booted to linux at the end of the ACS tests.

FVP instructions for ACS image and run

Download ACS image from:

- `https://gitlab.arm.com/systemready/acs/arm-systemready/-/tree/main/IR/prebuilt_images/v23.03_2.0.0`

Use the below command to run the FVP with EFI and ACS image support in the SD cards.

```
unxz ${<path-to-img>/ir-acs-live-image-generic-arm64.wic.xz}

<_workspace>/meta-arm/scripts/runfvp  --terminals=xterm <_workspace>/build/tmp/deploy/
↪images/corstone1000-fvp/corstone1000-image-corstone1000-fvp.fvpconf -- -C board.msd_
↪mmc.p_mmc_file=<path-to-img>/ir-acs-live-image-generic-arm64.wic -C board.msd_mmc_2.p_
↪mmc_file="${<path-to-img>/corstone1000-efi-partition.img}"
```

The test results can be fetched using following commands:

```
sudo mkdir /mnt/test
sudo mount -o rw,offset=<offset_3rd_partition> <path-to-img>/ir-acs-live-image-generic-
↪arm64.wic /mnt/test/
fdisk -lu <path-to-img>/ir-acs-live-image-generic-arm64.wic
->  Device                                               Start     End Sectors ␣
↪Size Type
     <path-to-img>/ir-acs-live-image-generic-arm64.wic1    2048  206847  204800   100M␣
↪Microsoft basic data
     <path-to-img>/ir-acs-live-image-generic-arm64.wic2  206848 1024239  817392 399.1M␣
↪Linux filesystem
     <path-to-img>/ir-acs-live-image-generic-arm64.wic3 1026048 1128447  102400    50M␣
↪Microsoft basic data
```

(continues on next page)

(continued from previous page)

```

->   <offset_3rd_partition> = 1026048 * 512 (sector size) = 525336576
```

The FVP will reset multiple times during the test, and it might take up to 1 day to finish the test. At the end of test, the FVP host terminal will halt showing a shell prompt. Once test is finished, the FVP can be stoped, and result can be copied following above instructions.

NOTE: A rare issue has been noticed (5-6% occurence) during which the FVP hangs during booting the system while running ACS tests. If this happens, please apply the following patch, rebuild the software stack for FVP and re-run the ACS tests.

```
cd <_workspace>
git clone https://git.gitlab.arm.com/arm-reference-solutions/systemready-patch.git -b␣
↪CORSTONE1000-2023.11
cp -f systemready-patch/embedded-a/corstone1000/sr_ir_workaround/0001-embedded-a-
↪corstone1000-sr-ir-workaround.patch meta-arm
cd meta-arm
git am 0001-embedded-a-corstone1000-sr-ir-workaround.patch
cd ..
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml -c="bitbake u-boot -c␣
↪cleanall; bitbake trusted-firmware-a -c cleanall; corstone1000-image -c cleanall;␣
↪bitbake corstone1000-image"
```

Common to FVP and FPGA

U-Boot should be able to boot the grub bootloader from the 1st partition and if grub is not interrupted, tests are executed automatically in the following sequence:

- SCT
- UEFI BSA
- FWTS

The results can be fetched from the `acs_results` folder in the RESULT partition of the USB stick (FPGA) / SD Card (FVP).

2.11 Manual capsule update and ESRT checks

The following section describes running manual capsule update.

The steps described in this section perform manual capsule update and show how to use the ESRT feature to retrieve the installed capsule details.

For the following tests two capsules are needed to perform 2 capsule updates. A positive update and a negative update.

A positive test case capsule which boots the platform correctly until the Linux prompt, and a negative test case with an incorrect capsule (corrupted or outdated) which fails to boot to the host software.

Check the "Run SystemReady-IR ACS tests" section above to download and unpack the ACS image file

- `ir-acs-live-image-generic-arm64.wic.xz`

Download u-boot under <_workspace> and install tools:

```
git clone https://github.com/u-boot/u-boot.git
cd u-boot
git checkout 83aa0ed1e93e1ffac24888d98d37a5b04ed3fb07
make tools-only_defconfig
make tools-only
```

NOTE: The following error could happen if the linux build system does not have “libgnutls28-dev”.

error: “tools/mkeficapsule.c:21:10: fatal error: gnutls/gnutls.h: No such file or directory”. If that’s the case please install libgnutls28-dev and its dependencies by using the following command.

```
sudo apt-get install -y libgnutls28-dev
```

Download systemready-patch repo under <_workspace>:

```
git clone https://git.gitlab.arm.com/arm-reference-solutions/systemready-patch.git -b␣
↪CORSTONE1000-2023.11
```

2.11.1 Generating Capsules

Generating FPGA Capsules

```
cd <_workspace>/build/tmp/deploy/images/corstone1000-mps3/
sh <_workspace>/systemready-patch/embedded-a/corstone1000/capsule_gen/capsule_gen.sh -d␣
↪mps3
```

This will generate a file called “corstone1000_image.nopt” which will be used to generate a UEFI capsule.

```
cd <_workspace>

./u-boot/tools/mkeficapsule --monotonic-count 1 --private-key build/tmp/deploy/images/
↪corstone1000-mps3/corstone1000_capsule_key.key \
--certificate build/tmp/deploy/images/corstone1000-mps3/corstone1000_capsule_cert.crt --
↪index 1 --guid 989f3a4e-46e0-4cd0-9877-a25c70c01329 \
--fw-version 6 build/tmp/deploy/images/corstone1000-mps3/corstone1000_image.nopt cs1k_
↪cap_mps3_v6

./u-boot/tools/mkeficapsule --monotonic-count 1 --private-key build/tmp/deploy/images/
↪corstone1000-mps3/corstone1000_capsule_key.key \
--certificate build/tmp/deploy/images/corstone1000-mps3/corstone1000_capsule_cert.crt --
↪index 1 --guid 989f3a4e-46e0-4cd0-9877-a25c70c01329 \
--fw-version 5 build/tmp/deploy/images/corstone1000-mps3/corstone1000_image.nopt cs1k_
↪cap_mps3_v5
```

Generating FVP Capsules

```
cd <_workspace>/build/tmp/deploy/images/corstone1000-fvp/
sh <_workspace>/systemready-patch/embedded-a/corstone1000/capsule_gen/capsule_gen.sh -d␣
↪fvp
```

This will generate a file called "corstone1000_image.nopt" which will be used to generate a UEFI capsule.

```
cd <_workspace>
./u-boot/tools/mkeficapsule --monotonic-count 1 --private-key build/tmp/deploy/images/
↪corstone1000-fvp/corstone1000_capsule_key.key \
--certificate build/tmp/deploy/images/corstone1000-fvp/corstone1000_capsule_cert.crt --
↪index 1 --guid 989f3a4e-46e0-4cd0-9877-a25c70c01329 \
--fw-version 6 build/tmp/deploy/images/corstone1000-fvp/corstone1000_image.nopt cs1k_cap_
↪fvp_v6

./u-boot/tools/mkeficapsule --monotonic-count 1 --private-key build/tmp/deploy/images/
↪corstone1000-fvp/corstone1000_capsule_key.key \
--certificate build/tmp/deploy/images/corstone1000-fvp/corstone1000_capsule_cert.crt --
↪index 1 --guid 989f3a4e-46e0-4cd0-9877-a25c70c01329 \
--fw-version 5 build/tmp/deploy/images/corstone1000-fvp/corstone1000_image.nopt cs1k_cap_
↪fvp_v5
```

Common Notes for FVP and FPGA

The capsule binary size (wic file) should be less than 15 MB.

Based on the user's requirement, the user can change the firmware version number given to `--fw-version` option (the version number needs to be >= 1).

2.11.2 Copying Capsules

Copying the FPGA capsules

The user should prepare a USB stick as explained in ACS image section *FPGA instructions for ACS image*. Place the generated `cs1k_cap` files in the root directory of the boot partition in the USB stick. Note: As we are running the direct method, the `cs1k_cap` file should not be under the EFI/UpdateCapsule directory as this may or may not trigger the on disk method.

```
sudo cp cs1k_cap_mps3_v6 <mounting path>/BOOT/
sudo cp cs1k_cap_mps3_v5 <mounting path>/BOOT/
sync
```

Copying the FVP capsules

First, mount the IR image:

```
sudo mkdir /mnt/test
sudo mount -o rw,offset=1048576 <path-to-img>/ir-acs-live-image-generic-arm64.wic  /mnt/
↪test
```

Then, copy the capsules:

```
sudo cp cs1k_cap_fvp_v6 /mnt/test/
sudo cp cs1k_cap_fvp_v5 /mnt/test/
sync
```

Then, unmount the IR image:

```
sudo umount /mnt/test
```

NOTE: Please refer to *FVP instructions for ACS image and run* section to find the first partition offset.

2.11.3 Performing the capsule update

During this section we will be using the capsule with the higher version (cs1k_cap_<fvp/mps3>_v6) for the positive scenario and the capsule with the lower version (cs1k_cap_<fvp/mps3>_v5) for the negative scenario.

Running the FVP with the IR prebuilt image

Run the FVP with the IR prebuilt image:

```
<_workspace>/meta-arm/scripts/runfvp --terminals=xterm <_workspace>/build/tmp/deploy/
↪images/corstone1000-fvp/corstone1000-image-corstone1000-fvp.fvpconf -- -C board.msd_
↪mmc.p_mmc_file=<path-to-img>/ir-acs-live-image-generic-arm64.wic
```

NOTE: <path-to-img> must start from the root directory. make sure there are no spaces before or after of "=". board.msd_mmc.p_mmc_file=<path-to-img>/ir-acs-live-image-generic-arm64.wic.

Running the FPGA with the IR prebuilt image

Insert the prepared USB stick then Power cycle the MPS3 board.

Executing capsule update for FVP and FPGA

Reach u-boot then interrupt the boot to reach the EFI shell.

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

Then, type FS0: as shown below:

```
FS0:
```

In case of the positive scenario run the update with the higher version capsule as shown below:

```
EFI/BOOT/app/CapsuleApp.efi cs1k_cap_<fvp/mps3>_v6
```

After successfully updating the capsule the system will reset.

In case of the negative scenario run the update with the lower version capsule as shown below:

```
EFI/BOOT/app/CapsuleApp.efi cs1k_cap_<fvp/mps3>_v5
```

The command above should fail and in the TF-M logs the following message should appear:

```
ERROR: flash_full_capsule: version error
```

Then, reboot manually:

```
Shell> reset
```

FPGA: Select Corstone-1000 Linux kernel boot

Remove the USB stick before u-boot is reached so the Corstone-1000 kernel will be detected and used for booting.

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

FVP: Select Corstone-1000 Linux kernel boot

Interrupt the u-boot shell.

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

Run the following commands in order to run the Corstone-1000 Linux kernel and being able to check the ESRT table.

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
```

2.11.4 Capsule update status

Positive scenario

In the positive case scenario, the user should see following log in TF-M log, indicating the new capsule image is successfully applied, and the board boots correctly.

```
...
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
...
```

It's possible to check the content of the ESRT table after the system fully boots.

In the Linux command-line run the following:

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

0x0
989f3a4e-46e0-4cd0-9877-a25c70c01329
0
6
0
6
0
```

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

Negative scenario (Applicable to FPGA only)

In the negative case scenario (rollback the capsule version), the user should see appropriate logs in the secure enclave terminal.

```
...
  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
...
```

If capsule pass initial verification, but fails verifications performed during boot time, secure enclave will try new images predetermined number of times (defined in the code), before reverting back to the previous good bank.

```
...
metadata_write: success: active = 0, previous = 1
```

(continues on next page)

(continued from previous page)

```
fwu_select_previous: in regular state by choosing previous active bank
...
```

It's possible to check the content of the ESRT table after the system fully boots.

In the Linux command-line run the following:

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

0x0
989f3a4e-46e0-4cd0-9877-a25c70c01329
0
6
1
5
0
```

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

Note: This test is currently not working properly in Corstone-1000 FVP. However, it is not part of the System-Ready IR tests, and it won't affect the SR-IR certification. All the compulsory capsule update tests for SR-IR works on both Corstone-1000 FVP and FPGA.

2.12 Linux distros tests

2.12.1 Debian install and boot preparation

There is a known issue in the Shim 15.7 provided with the Debian installer image (see below). This bug causes a fatal error when attempting to boot media installer for Debian, and it resets the platform before installation starts. A patch to be applied to the Corstone-1000 stack (only applicable when installing Debian) is provided to Skip the Shim. This patch makes U-Boot automatically bypass the Shim and run grub and allows the user to proceed with a normal installation. If at the moment of reading this document the problem is solved in the Shim, the user is encouraged to try the corresponding new installer image. Otherwise, please apply the patch as indicated by the instructions listed below. These instructions assume that the user has already built the stack by following the build steps of this documentation.

```
cd <_workspace>
git clone https://git.gitlab.arm.com/arm-reference-solutions/systemready-patch.git -b CORSTONE1000-2023.11
cp -f systemready-patch/embedded-a/corstone1000/shim/0001-arm-bsp-u-boot-corstone1000-Skip-the-shim-by-booting.patch meta-arm
```

(continues on next page)

(continued from previous page)

```
cd meta-arm
git am 0001-arm-bsp-u-boot-corstone1000-Skip-the-shim-by-booting.patch
cd ..
```

On FPGA

```
kas shell meta-arm/kas/corstone1000-mps3.yml:meta-arm/ci/debug.yml -c="bitbake u-boot␣
↪trusted-firmware-a corstone1000-image -c cleansstate; bitbake corstone1000-image"
```

On FVP

```
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml -c="bitbake u-boot␣
↪trusted-firmware-a corstone1000-image -c cleansstate; bitbake corstone1000-image"
```

On FPGA, please update the cs1000.bin on the SD card with the newly generated wic file.

NOTE: Skip the shim patch only applies to Debian installation. The user should remove the patch from meta-arm before running the software to boot OpenSUSE or executing any other tests in this user guide. You can make sure of removing the skip the shim patch by executing the steps below.

```
cd <_workspace>/meta-arm
git reset --hard HEAD~1
cd ..
kas shell meta-arm/kas/corstone1000-fvp.yml:meta-arm/ci/debug.yml -c="bitbake u-boot -c␣
↪cleanall; bitbake trusted-firmware-a -c cleanall; corstone1000-image -c cleanall;␣
↪bitbake corstone1000-image"
```

2.12.2 Preparing the Installation Media

Download one of following Linux distro images:

- Debian installer image (Tested on: debian-12.2.0-arm64-DVD-1.iso)
- OpenSUSE Tumbleweed installer image (Tested on: openSUSE-Tumbleweed-DVD-aarch64-Snapshot20231120-Media.iso)

NOTE: For OpenSUSE Tumbleweed, the user should look for a DVD Snapshot like openSUSE-Tumbleweed-DVD-aarch64-Snapshot<date>-Media.iso

FPGA

To test Linux distro install and boot on FPGA, the user should prepare two empty USB sticks (minimum size should be 4GB and formatted with FAT32).

The downloaded iso file needs to be flashed to your USB drive. This can be done with your development machine.

In the example given below, we assume the USB device is `/dev/sdb` (the user should use the *lsblk* command to confirm).

NOTE: Please don't confuse your host machine own hard drive with the USB drive. Then, copy the contents of the iso file into the first USB stick by running the following command in the development machine:

```
sudo dd if=<path-to-iso_file> of=/dev/sdb iflag=direct oflag=direct status=progress␣
↪bs=1M; sync;
```

FVP

To test Linux distro install and boot on FVP, the user should prepare an mmc image. With a minimum size of 8GB formatted with gpt.

```
#Generating mmc2
dd if=/dev/zero of=<_workspace>/mmc2_file.img bs=1 count=0 seek=8G; sync;
parted -s mmc2_file.img mklabel gpt
```

2.12.3 Debian/openSUSE install

FPGA

Unplug the first USB stick from the development machine and connect it to the MSP3 board. At this moment, only the first USB stick should be connected. Open the following picocom sessions in your development machine:

```
sudo picocom -b 115200 /dev/ttyUSB0  # in one terminal
sudo picocom -b 115200 /dev/ttyUSB2  # in another terminal.
```

When the installation screen is visible in ttyUSB2, plug in the second USB stick in the MPS3 and start the distro installation process. If the installer does not start, please try to reboot the board with both USB sticks connected and repeat the process.

NOTE: Due to the performance limitation of Corstone-1000 MPS3 FPGA, the distro installation process can take up to 24 hours to complete.

FVP

```
<_workspace>/meta-arm/scripts/runfvp --terminals=xterm <_workspace>/build/tmp/deploy/
↪images/corstone1000-fvp/corstone1000-image-corstone1000-fvp.fvpconf -- -C board.msd_
↪mmc.p_mmc_file="<path-to-iso_file>" -C board.msd_mmc_2.p_mmc_file="<_workspace>/mmc2_
↪file.img"
```

The installer should now start. The os will be installed on the second mmc 'mmc2_file.img'.

2.12.4 Debian install clarifications

As the installation process for Debian is different than the one for openSUSE, Debian may need some extra steps, that are indicated below:

During Debian installation, please answer the following question:

- "Force grub installation to the EFI removable media path?" Yes
- "Update NVRAM variables to automatically boot into Debian?" No

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.

2.12.5 Debian/openSUSE boot after installation

FPGA

Once the installation is complete, unplug the first USB stick and reboot the board. The board will then enter recovery mode, from which the user can access a shell after entering the password for the root user.

FVP

Once the installation is complete, you will need to exit the shell instance and run this command to boot into the installed OS:

```
<_workspace>/meta-arm/scripts/runfvp --terminals=xterm <_workspace>/build/tmp/deploy/
↪images/corstone1000-fvp/corstone1000-image-corstone1000-fvp.fvpconf -- -C board.msd_
↪mmc.p_mmc_file="<_workspace>/mmc2_file.img"
```

Once the FVP begins booting, you will need to quickly change the boot option in grub, to boot into recovery mode.

NOTE: This option will disappear quickly, so it’s best to preempt it.

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

Common

Proceed to edit the following files accordingly:

```
#Only applicable to Debian
vi /etc/systemd/system.conf
DefaultDeviceTimeoutSec=infinity
```

```
#Only applicable to openSUSE
vi /usr/lib/systemd/system.conf
DefaultDeviceTimeoutSec=infinity

The system.conf has been moved from /etc/systemd/ to /usr/lib/systemd/ and directly␣
↪modifying
the /usr/lib/systemd/system.conf is not working and it is getting overridden. We have to␣
↪create
```

(continues on next page)

(continued from previous page)

```
drop ins system configurations in /etc/systemd/system.conf.d/ directory. So, copy the
/usr/lib/systemd/system.conf to /etc/systemd/system.conf.d/ directory after the mentioned modifications.
```

The file to be edited next is different depending on the installed distro:

```
vi /etc/login.defs # Only applicable to Debian
vi /usr/etc/login.defs # Only applicable to openSUSE
LOGIN_TIMEOUT   180
```

To make sure the changes are applied, please run:

```
systemctl daemon-reload
```

After applying the previous commands, please reboot the board or restart the runfvp command.

The user should see a login prompt after booting, for example, for debian:

```
debian login:
```

Login with the username root and its corresponding password (already set at installation time).

NOTE: Debian/OpenSUSE Timeouts are not applicable for all systems. Some systems are faster than the others (especially when running the FVP) and works well with default timeouts. If the system boots to Debian or OpenSUSE unmodified, the user can skip this section.

2.13 PSA API tests

2.13.1 Run PSA API test commands (applicable to both FPGA and FVP)

When running PSA API test commands (aka PSA Arch Tests) on MPS3 FPGA, the user should make sure there is no USB stick connected to the board. Power on the board and boot the board to Linux. Then, the user should follow the steps below to run the tests.

When running the tests on the Corstone-1000 FVP, the user should follow the instructions in *Running the software on FVP* section to boot Linux in FVP host_terminal_0, and login using the username `root`.

First, load FF-A TEE kernel module:

```
insmod /lib/modules/*-yocto-standard/updates/arm-ffa-tee.ko
```

Then, check whether the FF-A TEE driver is loaded correctly by using the following command:

```
cat /proc/modules | grep arm_ffa_tee
```

The output should be:

```
arm_ffa_tee <ID> - - Live <address> (O)
```

Now, run the PSA API tests in the following order:

```
psa-iat-api-test
psa-crypto-api-test
```

(continues on next page)

(continued from previous page)

```
psa-its-api-test
psa-ps-api-test
```

NOTE: The psa-crypto-api-test takes between 30 minutes to 1 hour to run.

2.14 Tests results

As a reference for the end user, reports for various tests for Corstone-1000 software (CORSTONE1000-2023.11) can be found here.

2.15 Running the software on FVP on Windows or AArch64 Linux

The user should follow the build instructions in this document to build on a Linux host machine. Then, copy the output binaries to the Windows or Aarch64 Linux machine where the FVP is located. Then, launch the FVP binary.

2.16 Security Issue Reporting

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

CHAPTER

THREE

RELEASE NOTES

3.1 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.

3.2 Release notes - 2023.11

3.2.1 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.

3.2.2 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

3.3 Release notes - 2023.06

3.3.1 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.

3.3.2 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

3.4 Release notes - 2022.11.23

3.4.1 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.

3.4.2 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

3.5 Release notes - 2022.04.04

3.5.1 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

3.5.2 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

3.6 Release notes - 2022.02.25

3.6.1 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.

3.6.2 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

3.6.3 Release notes - 2022.02.21

3.6.4 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.

3.6.5 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

3.6.6 Release notes - 2022.01.18

3.6.7 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.

3.6.8 Release notes - 2021.12.15

3.6.9 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

3.6.10 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

3.6.11 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.

3.6.12 Release notes - 2021.10.29

3.6.13 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

3.6.14 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

3.6.15 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

3.6.16 Support

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

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

CHAPTER

FOUR

CHANGE LOG

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

4.1 Version 2023.11

4.1.1 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

4.1.2 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

4.1.3 Yocto distribution components versions

meta-arm	nanbield
poky	nanbield
meta-openembedded	nanbield
meta-secure-core	nanbield
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

4.2 Version 2023.06

4.2.1 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
- NVMXIP block storage support in U-Boot
- Upgrading the SW stack recipes
- Upgrades for the U-Boot FF-A driver and MM communication

4.2.2 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

4.2.3 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

4.3 Version 2022.11.23

4.3.1 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

4.3.2 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

4.3.3 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

4.4 Version 2022.04.04

4.4.1 Changes

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

4.5 Version 2022.02.25

4.5.1 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

4.6 Version 2022.02.21

4.6.1 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.

4.7 Version 2022.01.18

4.7.1 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

4.8 Version 2022.01.07

4.8.1 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.

4.9 Version 2021.12.15

4.9.1 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

4.10 Version 2021.10.29

4.10.1 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)