



STM32MP15 TF-A



# STM32MP15 TF-A

Stable: 21.02.2020 - 08:28 / Revision: 02.01.2020 - 13:51

A quality version of this page, [accepted](#) on 21 February 2020, was based off this revision.

## Contents

1 Article Purpose .....	2
2 Overview .....	2
3 Developer Package .....	3
<b>3.1 Install sources</b> .....	<b>3</b>
<b>3.2 Official source tree</b> .....	<b>3</b>
<b>3.3 Build Process</b> .....	<b>4</b>
3.3.1 Initialise the cross compile environment .....	4
3.3.2 TF-A Build flags .....	4
<b>3.4 Build command</b> .....	<b>5</b>
<b>3.5 Final image</b> .....	<b>5</b>
4 Distribution Package .....	5
<b>4.1 Access sources</b> .....	<b>5</b>
5 Update software on board .....	6
<b>5.1 Partitioning of binaries</b> .....	<b>6</b>
<b>5.2 Update via SDCARD</b> .....	<b>6</b>
<b>5.3 Update via USB mass storage on U-boot</b> .....	<b>7</b>
<b>5.4 Update your boot device (including SD card on the target)</b> .....	<b>7</b>

## 1 Article Purpose

This section details the process used to build TF-A from sources and to deploy it on your target. The build example is based on the OpenSTLinux environment:

- Developer Package
- Distribution Package

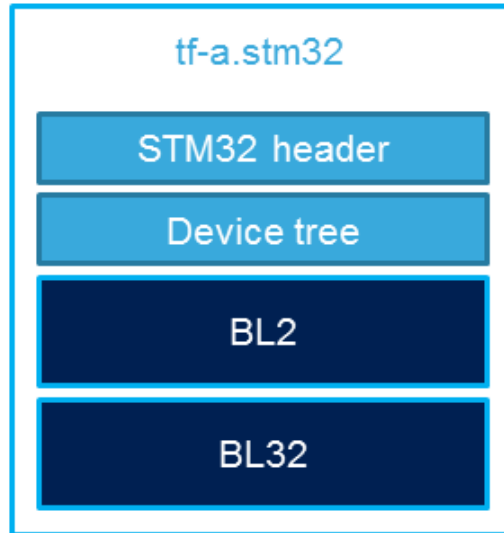
## 2 Overview

TF-A is the FSBL for the ST trusted boot chain. It must be configured or updated depending on your platform.

Cross compilation of TF-A is only required if it is to be modified. By default, in the Starter Package, the TF-A image is named: `tf-<board>-trusted.stm32`.

In changes are made, you must rebuild TF-A and update all the FSBL partitions of your boot device with this new image. A second FSBL image is used as a backup image.

The build process creates a full STM32 image that can be used for Flash integrating a specific header. This trusted firmware-A image contains a device tree, a BL2 and a BL32 stage. These binaries are built in a single step during the build process.



## 3 Developer Package

### 3.1 Install sources

The Developer Package contains OpenSTLinux and TF-A sources: [TF-A Installation](#)

### 3.2 Official source tree



**The STM32MP1 platform is not yet fully upstreamed. So features are not available from this repository**

Download source code from the official github

```
PC $> git clone https://github.com/ARM-software/arm-trusted-firmware.git
```

## 3.3 Build Process

### 3.3.1 Initialise the cross compile environment

Setup Cross compile environment

### 3.3.2 TF-A Build flags

For ecosystem release v1.2.0 

Mandatory flags:

- ARM\_ARCH\_MAJOR=7: the major version of ARM Architecture to target (STM32MP15 is ARMv7 architecture based)
- ARCH=aarch32: specify aarch32 architecture to be built
- PLAT=stm32mp1: builds an stm32mp1 platform
- DTB\_FILE\_NAME=<fdt file name>.dtb: this must be defined to build the proper target and include the correct DTB file into the final file
- AARCH32\_SP=<monitor>
  - sp\_min: builds the BL32 secure monitor if required
  - optee: do not include BL32 and prepare BL2 for optee-specific load which also requires to build with the 'STM32MP\_BOOT\_ONLY=1' option.

Optional flags:

- STM32MP\_BOOT\_ONLY=1: Build all storage driver supports except serial (no stm32\_programmer support). Required for OP-TEE mode.
- STM32MP\_FLASHLOADER\_ONLY=1: Build only serial link driver support (UART/USB) and stm32\_programmer support.
- DEBUG=1: add debug information in all binaries
- V=1: print verbose compilation traces

For ecosystem release v1.1.0 

Mandatory flags:

- ARM\_ARCH\_MAJOR=7: the major version of ARM Architecture to target (STM32MP15 is ARMv7 architecture based)
- ARCH=aarch32: specify aarch32 architecture to be built
- PLAT=stm32mp1: builds an stm32mp1 platform
- DTB\_FILE\_NAME=<fdt file name>.dtb: this must be defined to build the proper target and include the correct DTB file into the final file
- AARCH32\_SP=<monitor>
  - sp\_min: builds the BL32 secure monitor if required
  - optee: do not include BL32 and prepare BL2 for optee-specific load

Optional flags:

- DEBUG=1: add debug information in all binaries
- V=1: print verbose compilation traces

## 3.4 Build command

From the Developer Package tarball, a Makefile.sdk is present and must be used to build the target. It automatically sets the proper configuration for the TF-A build.

```
PC $> make -f Makefile.sdk TF_A_CONFIG=trusted TFA_DEVICETREE=stm32mp157c-<board>
```

The latest version of the helper file is also available in this user guide: [README.HOW\\_TO.txt](#).



**The DTB\_FILE\_NAME flag must be set to select the correct board configuration.**

**The device tree file for the target must be located in fdt's folder (stm32mp157c-<board>.dts)**

If no Makefile.sdk exists, you must add your own environment flags:

```
PC $> unset LDFLAGS;  
PC $> unset CFLAGS;  
PC $> make ARM_ARCH_MAJOR=7 ARCH=aarch32 PLAT=stm32mp1 AARCH32_SP=sp_min DTB_FILE_NAME  
=stm32mp157c-<board>.dtb
```

## 3.5 Final image

Final image is available for Flash or SD card update in the corresponding folder:

```
build/<target>/<debug|release>/tf-a-<target>.stm32  
Ex:  
build/stm32mp1/debug/tf-a-stm32mp157c-ev1.stm32
```

# 4 Distribution Package

For an OpenSTLinux distribution, the TF-A image is built in release mode by default. The yocto recipe can be found in:

```
meta-st/meta-st-stm32mp/recipes-bsp/trusted-firmware-a/tf-a-stm32mp_<version>.bb
```

If you want to modify the TF-A code source, use the following steps starting from an already downloaded and built OpenSTLinux distribution.

## 4.1 Access sources

You can use devtool to access the source.

```
PC $> cd <baseline root directory>  
PC $> devtool modify tf-a-stm32mp sources/boot/tf-a
```

By going to the sources/boot/tf-a folder, you can manage and modify the TF-A sources. To rebuild it, go back to the build-  
<distribution> folder and launch the TF-A recipe:

```
PC $> bitbake tf-a-stm32mp
```

The final image is deployed in the image default output folder.

## 5 Update software on board

### 5.1 Partitioning of binaries

The TF-A build provides a binary named tf-a-stm32mp157c-<board>.stm32 that MUST be copied to a dedicated partition named "fsblX" (X depends of needed backup).



**TF-A must be located in the first partition of your boot device.**

You can just update the first partition for a simple test, but all backup partitions must contain the same image at the end.

### 5.2 Update via SDCARD

If you use an SD card, you can simply update TF-A using the dd command on your host.

Plug your SD card into the computer and copy the binary to the dedicated partition; on an SDCard/USB disk the "fsbl1" partition is partition 1:

- SDCARD: /dev/mmcblkXp1 (where X is the instance number)
- SDCARD via USB reader: /dev/sdX1 (where X is the instance number)

- Linux

```
PC $> dd if=<tf-a file> of=/dev/<device partition> bs=1M conv=fdatasync
```

To find the partition associated to a specific label, just plug the SDCARD/USB disk into your PC and call the following command:

```
PC $> ls -l /dev/disk/by-partlabel/
total 0
lrwxrwxrwx 1 root root 10 Jan 17 17:38 bootfs -> ../../mmcblk0p4
lrwxrwxrwx 1 root root 10 Jan 17 17:38 fsbl1 -> ../..
/mmcblk0p1      -> FSBL1 (TF-A)
```





```
lrwxrwxrwx 1 root root 10 Jan 17 17:38 fsbl2 -> ../../  
/mmcblk0p2      → FSBL2 (TF-A backup – same content as FSBL)  
lrwxrwxrwx 1 root root 10 Jan 17 17:38 rootfs -> ../../mmcblk0p5  
lrwxrwxrwx 1 root root 10 Jan 17 17:38 ssbl -> ../../  
/mmcblk0p3      → SSBL (U-Boot)  
lrwxrwxrwx 1 root root 10 Jan 17 17:38 userfs -> ../../mmcblk0p6
```

- Windows

There is an existing dd for Windows that makes binary copying possible.

## 5.3 Update via USB mass storage on U-boot

See [How to use USB mass storage in U-Boot](#)

Follow the previous section to put `tf-a-<board>.stm32` onto SDCard/USB disk

## 5.4 Update your boot device (including SD card on the target)

Refer to the [STM32CubeProgrammer](#) documentation to update your target.

Trusted Firmware for Arm Cortex-A

First Stage Boot Loader

Boot Loader stage 2

Boot Loader stage 3-2

Device Tree Binary (or Blob)

Open Portable Trusted Execution Environment

Universal Asynchronous Receiver/Transmitter

SD memory card (<https://www.sdcard.org>)

Second Stage Boot Loader

Das U-Boot -- the Universal Boot Loader (see [U-Boot\\_overview](#))