



Example of directory structure for Packages

Example of directory structure for Packages



A quality version of this page, accepted on *31 March 2021*, was based off this revision.

Contents

1 Article purpose	3
2 Creating the structure	4
3 Focus on the Starter Package directory	5
4 Focus on the Developer Package directory	7
5 Focus on the Distribution Package directory	10
6 Appendix A: directory structure after build (Developer Package)	12
7 Appendix B: directory structure after build (Distribution Package)	15



1 Article purpose

This article aims at proposing a way to organize, on the host PC, the software packages of the different Packages (Starter, Developer and Distribution) for a given release of the STM32MPU Embedded Software distribution.

The main objective of the proposed organization is to keep together the software packages corresponding to a given release because there are links between them. For example:

- Flashing the image from the Starter Package on the board is mandatory before modifying the source code from the Developer Package. Both the image and the source code must belong to the same software release.
- The SDK (Developer Package) and the image (Starter Package) have both been generated from the Distribution Package. A software release thus guarantees that there is no misalignment between the different software packages.



The objective of this article is to describe one organization among all the possible organizations. Feel free to organize the delivered Packages in any other way that would better match your way of working.



In practice, this article uses the release **STM32MP15-Ecosystem-v3.0.0** for the STM32MPU Embedded Software distribution as an example to illustrate the proposed organization. If you are using a different release, the names of the directories and files might differ.

The directories are shown in green, while the files are in black.



2 Creating the structure

- Create your <working directory> and assign a unique name to it (for example by including the release name):

```
PC $> mkdir STM32MP15-Ecosystem-v3.0.0
PC $> cd STM32MP15-Ecosystem-v3.0.0
```

- Create the first-level directories that will host the software packages delivered through the STM32MPU Embedded Software distribution release note:

```
PC $> mkdir Starter-Package
PC $> mkdir Developer-Package
PC $> mkdir Distribution-Package
```

- The resulting directory structure looks as follows:

```
STM32MP15-Ecosystem-v3.0.0
├── Developer-Package
├── Distribution-Package
└── Starter-Package
```

STM32MPU Embedded Software release
Developer Package installation directory
Distribution Package installation directory
Starter Package installation directory

Once all software packages have been installed according to the instructions given in the STM32MPU Embedded Software distribution release note, the resulting directory structure looks as follows:

```
STM32MP15-Ecosystem-v3.0.0
├── Developer-Package
│   ├── SDK
│   ├── STM32Cube_FW_MP1_V1.4.0
│   └── stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31
│       ├── TEE OS source code (OpenSTLinux distribution)
│       └── Distribution-Package
│           ├── stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31
│           └── Starter-Package
│               ├── source code and OpenEmbedded-based build framework
│               └── stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31
│                   ├── OpenSTLinux distribution (full
│                   │   source code and OpenEmbedded-based build framework)
│                   └── Starter Package installation
│                       ├── Software image (binaries)
```

STM32MPU Embedded Software release
Developer Package installation
SDK for OpenSTLinux distribution
STM32CubeMP1 Package
Linux kernel, U-Boot, TF-A and OP-
TEE OS source code (OpenSTLinux distribution)
Distribution Package installation
OpenSTLinux distribution (full
source code and OpenEmbedded-based build framework)
Starter Package installation
Software image (binaries)



3 Focus on the Starter Package directory

The *Starter-Package* directory contains the software image for the STM32MPU Embedded Software distribution.

The trusted boot chain is the default solution delivered by STMicroelectronics. It includes the superset of features (for example, all Flash memory devices are supported). The basic boot chain is also upstreamed by STMicroelectronics, with a limited number of features (for example microSD card memory boot only). Refer to the [Boot chain overview](#) article for details.

Flash memory partitions (e.g. rootfs, bootfs...) are explained in the [STM32MP15 Flash mapping](#) article.

```

Starter-Package
├── stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31
│   └── images
│       └── stm32mp1
│           ├── flashlayout_st-image-weston                    Flash layout
│           └── files (description of the partitions) for the supported Flash devices and boards
│               ├── FlashLayout_emmc_stm32mp157c-ev1-optee.tsv          Flash layout
│               └── file for eMMC and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│                   ├── FlashLayout_emmc_stm32mp157c-ev1-trusted.tsv    Flash layout
│                   └── file for eMMC and trusted boot chain → STM32MP157C-EV1
│                       ├── FlashLayout_sdcard_stm32mp157c-dk2-basic.tsv  Flash layout
│                       └── file for microSD card and basic boot chain → STM32MP157C-DK2
│                           ├── FlashLayout_sdcard_stm32mp157c-dk2-optee.tsv  Flash layout
│                           └── file for microSD card and trusted boot chain, with OP-TEE OS → STM32MP157C-DK2
│                               ├── FlashLayout_sdcard_stm32mp157c-dk2-trusted.tsv  Flash layout
│                               └── file for microSD card and trusted boot chain (recommended setup) → STM32MP157C-DK2
│                                   ├── FlashLayout_sdcard_stm32mp157c-dk2-extensible.tsv  Flash layout
│                                   └── file for microSD card with no userfs partition but a rootfs partition extended to sdcard
│                                       size (recommended setup for package repository service) → STM32MP157C-DK2
│                                       └── [...]
│                                           ├── fip
│                                           │   ├── fip-<board>-<boot-type>.bin          Image to flash
│                                           │   └── arm-trusted-firmware
│                                           │       └── tf-a-<board>-<boot-type>.stm32      Intermediate tf-a binary used
│                                           └── to build fip image
│                                               ├── u-boot
│                                               │   ├── u-boot-nodtb-soc.bin          Intermediate tf-a binary used
│                                               │   └── to build fip image
│                                               └── build fip image
│                                                   ├── u-boot-<board>-<boot-type>.dtb      Intermediate tf-a dtb used to
│                                                   └── partition
│                                                       ├── optee
│                                                       │   ├── tee-header_v2-<board>.bin
│                                                       │   ├── tee-pageable_v2--<board>.bin
│                                                       │   └── tee-pager_v2--<board>.bin
│                                                       ├── kernel
│                                                       │   └── vmlinux                                vmlinux copied in bootfs
│                                                       └── scripts
│                                                           ├── create_sdcard_from_flashlayout.sh
│                                                           └── tfs partition
│                                                               ├── st-image-bootfs-openstlinux-weston-stm32mp1.ext4    Binary for boo
│                                                               └── rfs partition
│                                                                   ├── st-image-bootfs-openstlinux-weston-stm32mp1.manifest
│                                                                   └── rfs partition
│                                                                       ├── st-image-userfs-openstlinux-weston-stm32mp1.ext4    Binary for use
│                                                                       └── dorfs partition
│                                                                           ├── st-image-userfs-openstlinux-weston-stm32mp1.manifest
│                                                                           └── dorfs partition
│                                                                               ├── st-image-vendorfs-openstlinux-weston-stm32mp1.ext4    Binary for ven
│                                                                               └── dorfs partition
│                                                                                   ├── st-image-weston-openstlinux-weston-stm32mp1.ext4
│                                                                                   └── Binary for roo

```



***tfs* partition**

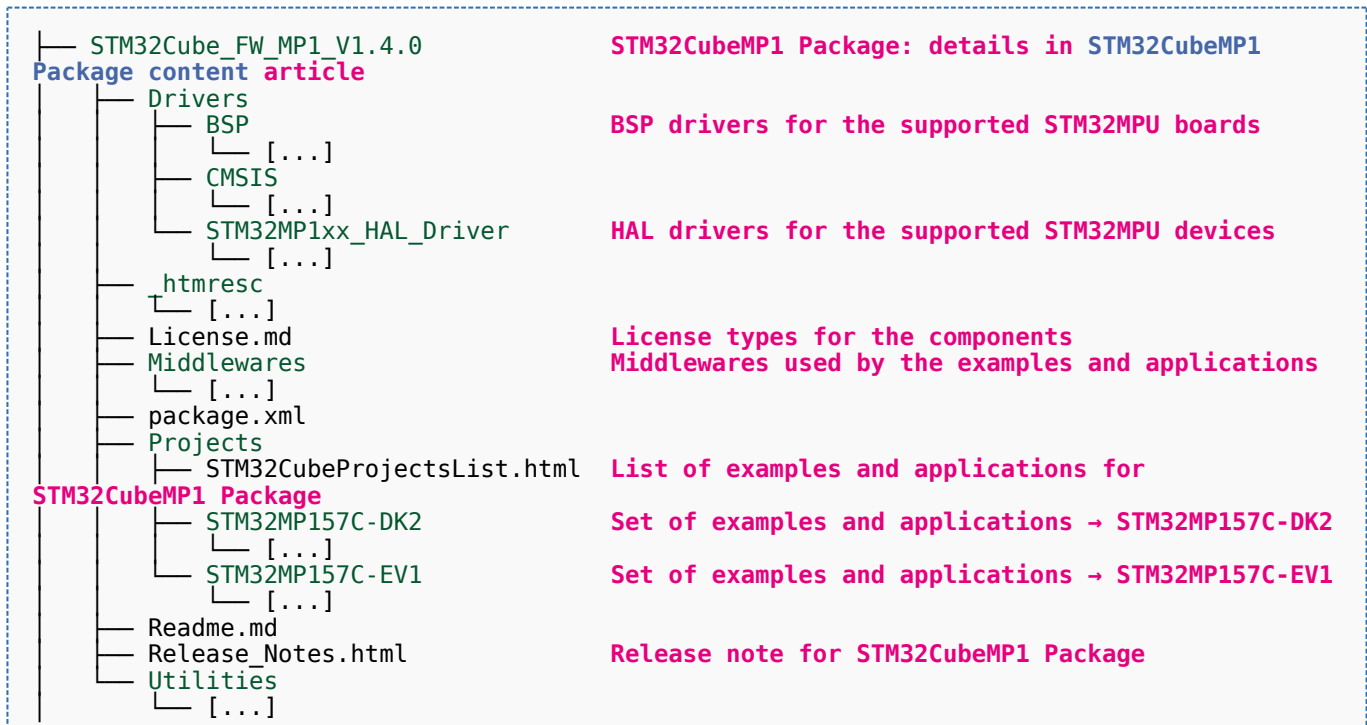
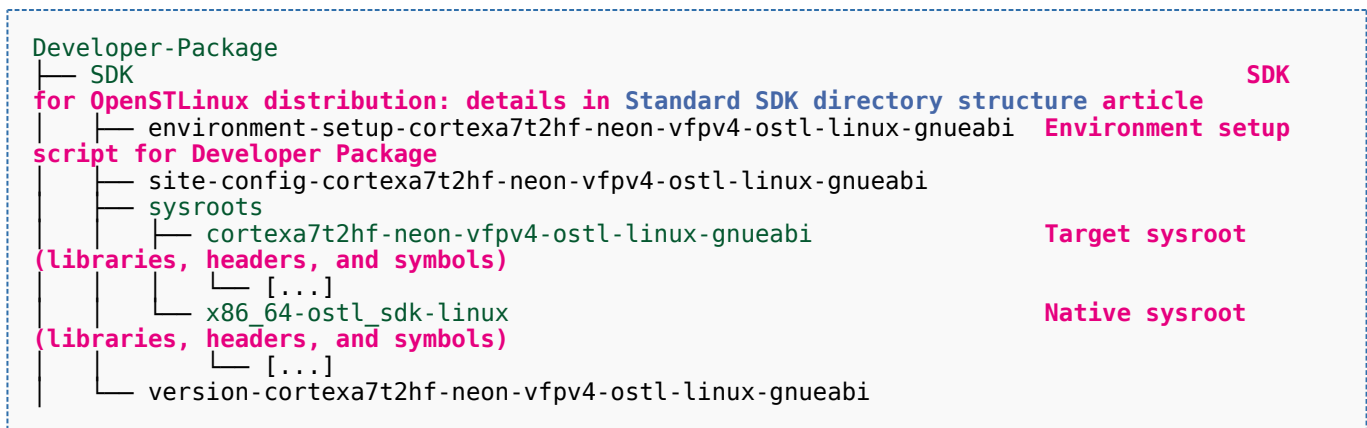
- st-image-weston-openstlinux-weston-stm32mp1.license
- st-image-weston-openstlinux-weston-stm32mp1-license_content.html
- st-image-weston-openstlinux-weston-stm32mp1.manifest
- [...]



4 Focus on the Developer Package directory

The *Developer-Package* directory contains:

- The source code for the following OpenSTLinux software packages (development for Arm®Cortex®-A processor):
 - Linux® kernel
 - U-Boot
 - TF-A
 - OP-TEE OS
- The debug symbol files for Linux kernel, U-Boot, TF-A and OP-TEE OS
- The SDK (for cross-development on an host PC)
- The STM32Cube MPU Package (developed for Arm®Cortex®-M processor)





Example of directory structure for Packages

```

└─ stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31
  distribution
  └─ images
    └─ stm32mp1
      directory
      └─ tf-a-bl2-optee.elf      Source code for OpenSTLinux
      TEE OS → trusted boot firmware stage
      └─ tf-a-bl2-trusted.elf  Debug symbol files installation
      boot firmware stage
      └─ tf-a-bl32-trusted.elf  Debug symbol file for TF-A, with OP-
      software stage          Debug symbol file for TF-A → trusted
      TEE OS → STM32MP157A-DK1  Debug symbol file for TF-A → runtime
      └─ u-boot-stm32mp157a-dk1-optee.elf  Debug symbol file for U-Boot, with OP-
      STM32MP157A-DK1          Debug symbol file for U-Boot →
      └─ u-boot-stm32mp157a-dk1-trusted.elf
      TEE OS → STM32MP157C-DK2  Debug symbol file for U-Boot, with OP-
      └─ u-boot-stm32mp157c-dk2-optee.elf  Debug symbol file for U-Boot →
      STM32MP157C-DK2          Debug symbol file for U-Boot, with OP-
      └─ u-boot-stm32mp157c-dk2-trusted.elf
      TEE OS → STM32MP157C-EV1  Debug symbol file for U-Boot →
      └─ u-boot-stm32mp157c-ev1-optee.elf  Debug symbol file for U-Boot, with OP-
      STM32MP157C-EV1          Debug symbol file for U-Boot →
      └─ u-boot-stm32mp157c-ev1-trusted.elf
      vmlinux                  Debug symbol file for Linux kernel
      [...]

```

```

└─ sources
  └─ arm-ostl-linux-gnueabi
    └─ FIP artifacts
      fip/                      fip images of Stater Package
      arm-trusted-firmware/    intermediate tf-a binaries from Stater Package
      (needed to build any fip image)
      optee/                    intermediate optee binaries from Stater Package
      (needed to build any fip image)
      u-boot/                   intermediate u-boot binaries from Stater Package
      (needed to build any fip image)

```

```

└─ linux-5.10.10
  └─ [*].patch                 ST patches for Linux kernel
  └─ fragment-[*].config      ST configuration fragments for Linux kernel
  └─ linux-5.10.10            Linux kernel source code directory
  └─ linux-5.10.10.tar.xz
  └─ README.HOW_TO.txt       Helper file for Linux kernel management: referenc
  e for Linux kernel build
  └─ series

```

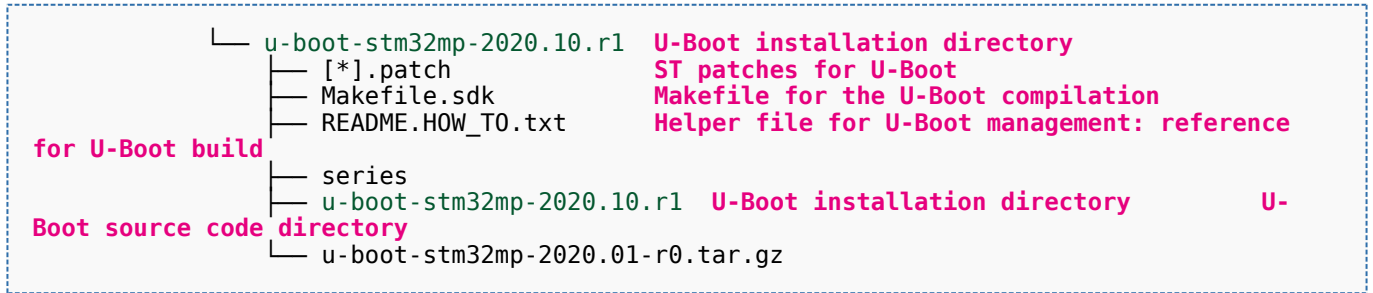
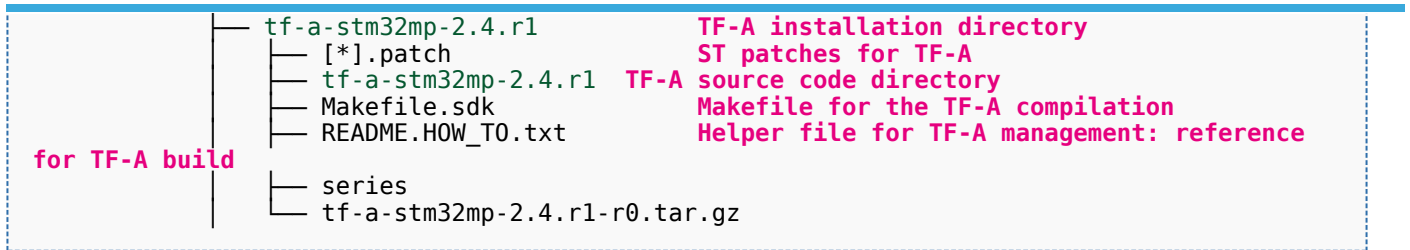
```

└─ optee-os-stm32mp-3.12.0.r1  OP-TEE OS installation directory
  └─ [*].patch                 ST patches for OP-TEE OS
  └─ optee-os-stm32mp-3.12.0.r1
  └─ Makefile.sdk             Makefile for the OP-TEE OS compilation
  └─ optee-os-stm32mp-3.12.0.r1-r0.tar.gz  OP-TEE OS source code
  directory
  └─ README.HOW_TO.txt       Helper file for OP-TEE OS management: reference
  for OP-TEE OS build
  └─ series

```




Example of directory structure for Packages



Appendix A shows the structure of the Linux kernel, U-Boot, TF-A and OP-TEE OS installation directories after these software packages have been built.



5 Focus on the Distribution Package directory

The *Distribution-Package* directory contains all the OpenEmbedded layers required to get the source code of any STM32MPU Embedded Software component, as well as a build framework based on OpenEmbedded.

```

Distribution-Package
├── openstlinux-5.10-dunfell-mp1-21-03-31  OpenSTLinux distribution
│   └── layers
│       ├── meta-openembedded  Collection of layers for the OpenEmbedded-Core universe (Op
OpenEmbedded standard)
│       │   └── [...]
│       └── meta-qt5           QT5 layer for OpenEmbedded (standard)
│           └── [...]

```

```

├── meta-st
│   ├── meta-st-openstlinux  STMicroelectronics layer that
contains the settings of the frameworks and images for the OpenSTLinux distribution
│   ├── recipes-st
│   │   ├── images
│   │   │   └── st-image-core.bb  Core image for OpenSTLinux
│   │   └── st-image-weston.bb  Weston image with basic Wayland
│   │                               support for OpenSTLinux distribution: recommended setup
│   └── [...]
│       ├── packagegroups
│       │   └── [...]

```

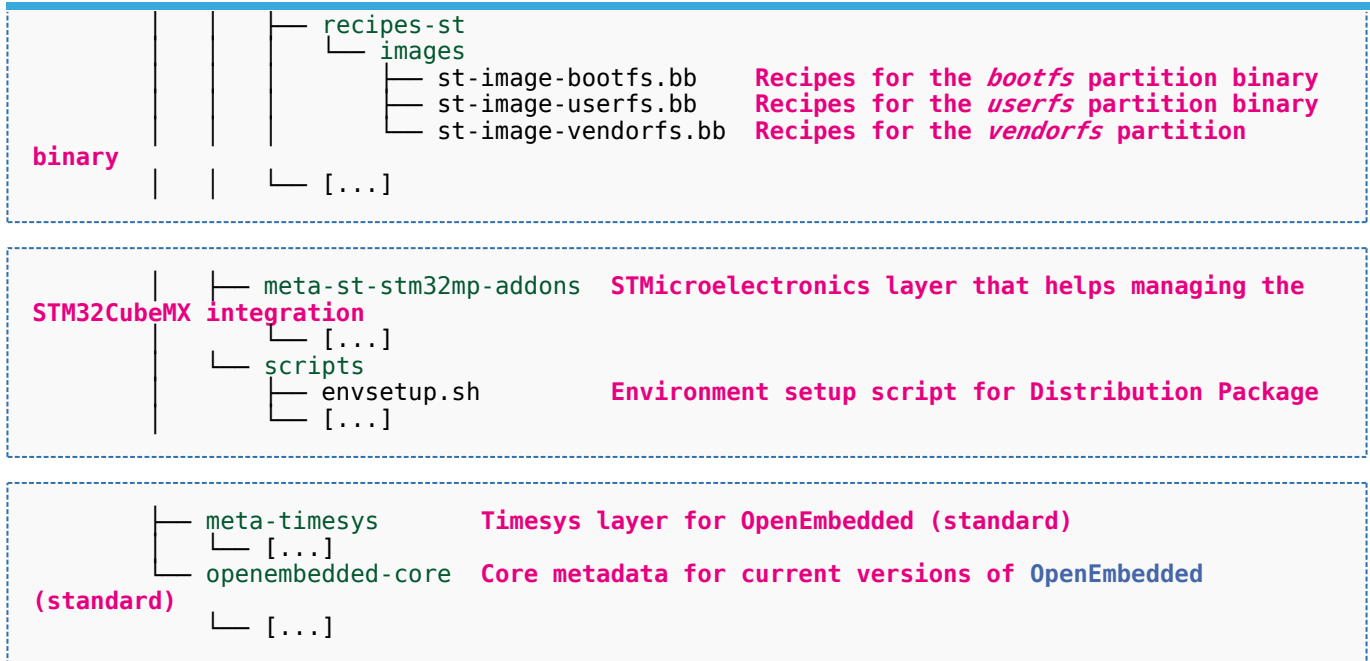
```

├── meta-st-stm32mp  STMicroelectronics layer that contains
the description of the BSP for the STM32 MPU devices
├── recipes-bsp
│   ├── alsa  Recipes for ALSA control configuration
│   └── drivers  Recipes for Vivante GCNANO GPU kernel
├── [...]
├── trusted-firmware-a  Recipes for TF-A
├── u-boot  Recipes for U-Boot
├── recipes-extended  Recipes for STM32Cube MPU Package
├── m4projects
│   ├── [...]
│   └── [...]
├── recipes-graphics  Recipes for Vivante libraries OpenGL
├── gcnano-userland
│   ├── [...]
│   └── [...]
├── recipes-kernel  Recipes for Linux kernel
├── linux
│   ├── [...]
│   └── linux-firmware  Recipes for Linux firmwares (example,
Bluetooth firmware)
└── [...]

```



Example of directory structure for Packages

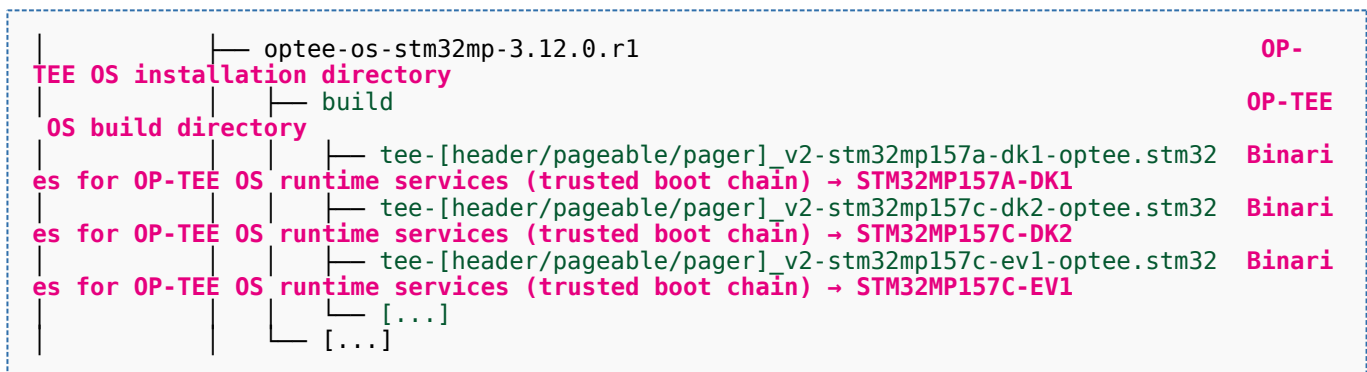
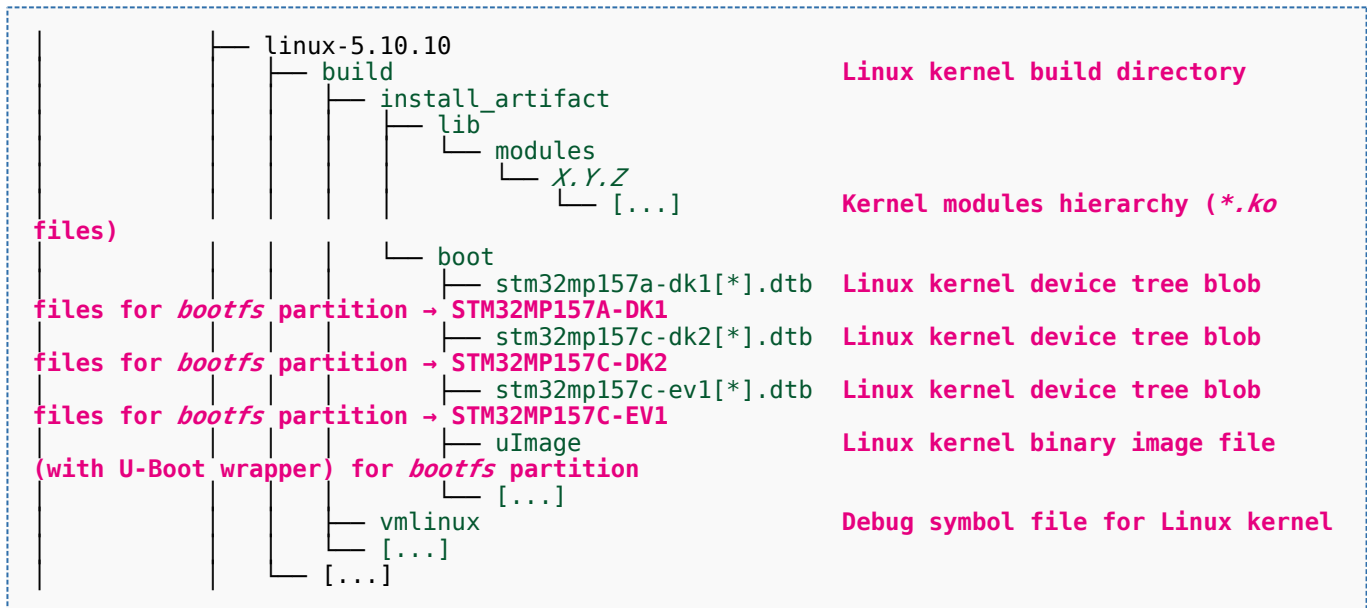
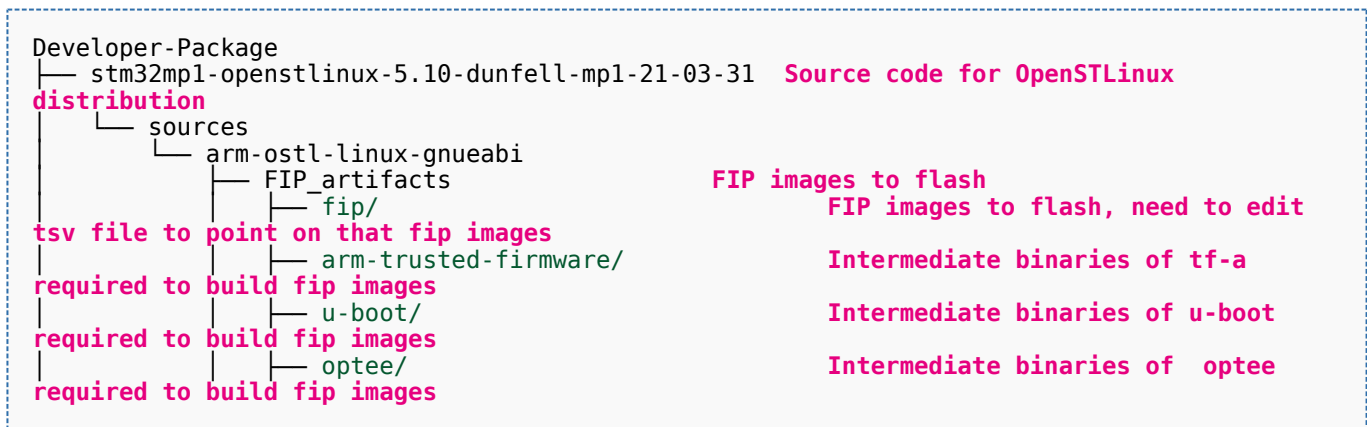


Appendix B shows the structure of the build directory.



6 Appendix A: directory structure after build (Developer Package)

Provided you have followed the recommendations of the *README.HOW_TO.txt* helper files to build the Linux kernel, the U-Boot and the TF-A, then the following new directories and files contain the build outputs.





Example of directory structure for Packages

tf-a-stm32mp-2.4.r1	TF-A installation directory
build	TF-A build directory
optee	TF-A, with OP-TEE OS
tf-a-bl2-optee.elf	Debug symbol file for TF-A
tf-a-stm32mp157a-dk1-optee.stm32	TF-A binary for <i>FSBL</i>
tf-a-stm32mp157c-dk2-optee.stm32	TF-A binary for <i>FSBL</i>
tf-a-stm32mp157c-ev1-optee.stm32	TF-A binary for <i>FSBL</i>
[...]	
trusted	TF-A, without OP-TEE OS
tf-a-bl2-trusted.elf	Debug symbol file for TF-A
tf-a-bl32-trusted.elf	Debug symbol file for TF-A
tf-a-stm32mp157a-dk1-trusted.stm32	TF-A binary for <i>FSBL</i>
tf-a-stm32mp157c-dk2-trusted.stm32	TF-A binary for <i>FSBL</i>
tf-a-stm32mp157c-ev1-trusted.stm32	TF-A binary for <i>FSBL</i>
[...]	

u-boot-stm32mp-2020.10.r1	U-Boot installation
build-basic	U-Boot build directory
u-boot-spl.stm32-stm32mp157a-dk1-basic	U-Boot binary for <i>FSBL</i>
u-boot-spl.stm32-stm32mp157c-dk2-basic	U-Boot binary for <i>FSBL</i>
u-boot-spl.stm32-stm32mp157c-ev1-basic	U-Boot binary for <i>FSBL</i>
u-boot-stm32mp157a-dk1-basic.img	U-Boot binary for <i>SSBL</i>
u-boot-stm32mp157c-dk2-basic.img	U-Boot binary for <i>SSBL</i>
u-boot-stm32mp157c-ev1-basic.img	U-Boot binary for <i>SSBL</i>
build-optee	U-Boot build directory
u-boot-stm32mp157a-dk1-optee.elf	Debug symbol file for U-Boot, with OP-TEE OS → STM32MP157A-DK1
u-boot-stm32mp157a-dk1-optee.stm32	U-Boot binary for <i>SSBL</i>
u-boot-stm32mp157c-dk2-optee.elf	Debug symbol file for U-Boot, with OP-TEE OS → STM32MP157C-DK2
u-boot-stm32mp157c-dk2-optee.stm32	U-Boot binary for <i>SSBL</i>
u-boot-stm32mp157c-ev1-optee.elf	Debug symbol file for U-Boot, with OP-TEE OS → STM32MP157C-EV1
u-boot-stm32mp157c-ev1-optee.stm32	U-Boot binary for <i>SSBL</i>
[...]	
build-trusted	U-Boot build directory
u-boot-stm32mp157a-dk1-trusted.elf	Debug symbol file for U-Boot → STM32MP157A-DK1
u-boot-stm32mp157a-dk1-trusted.stm32	U-Boot binary for <i>SSBL</i>



7 Appendix B: directory structure after build (Distribution Package)

Provided you have followed the build method explained in OpenSTLinux distribution, then the following new directories contain the build outputs.

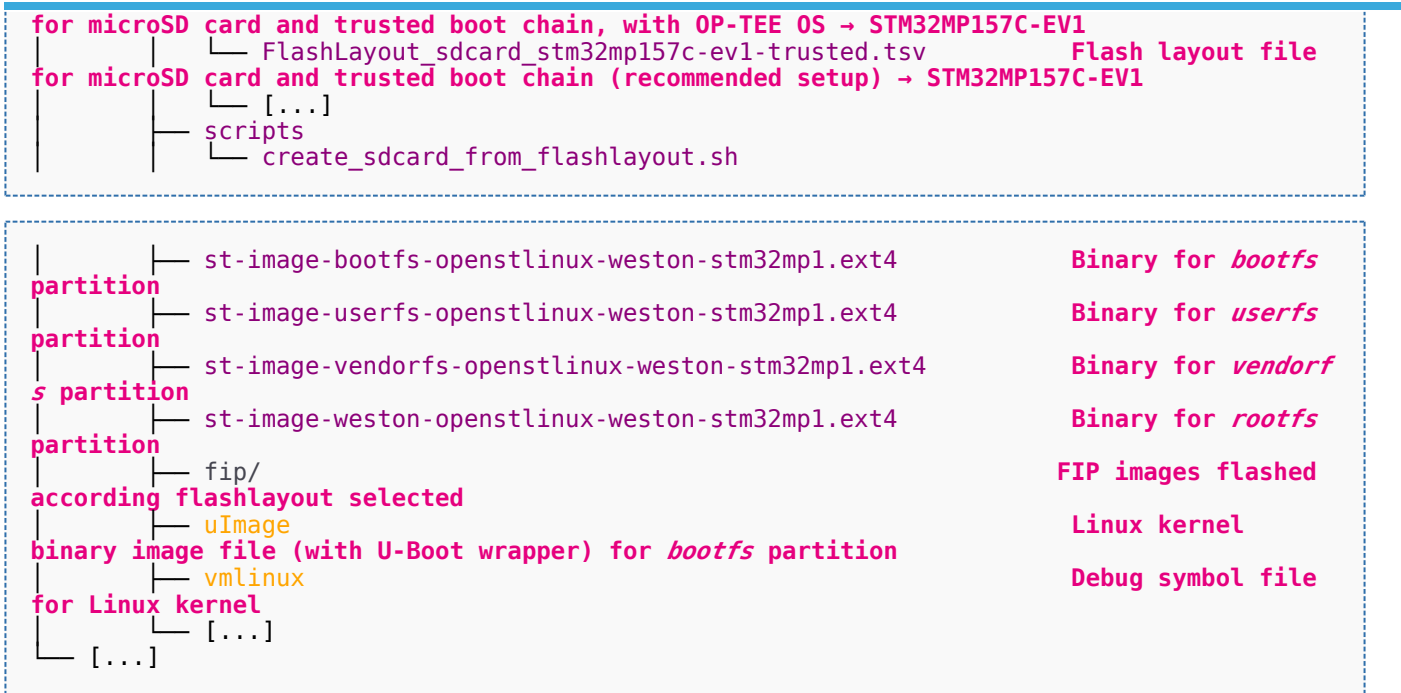
As long as you did not modify the source code:

- the files in **STPurple** are the same as the ones available in the **Starter Package**: flash layout, binaries for *bootfs*, *rootfs*, *userfs* and *vendorfs* partitions
- the files in grey are the same as the ones available in the **Starter and Developer Packages**: binaries for *FSBL* and *SSBL* partitions, and for OP-TEE OS runtime services
- the files in **orange** are the same as the ones available in the **Developer Package**: Linux kernel image and device tree blobs, and debug symbol files

```
Distribution-Package/openstlinux-5.10-dunfell-mp1-21-03-31 /build-openstlinuxweston-
stm32mp/tmp-glibc/deploy
├── images
│   └── stm32mp1
│       ├── flashlayout_st-image-weston                               Flash layout
│       └── files (description of the partitions) for the supported flash devices
│           ├── FlashLayout_emmc_stm32mp157c-ev1-optee.tsv           Flash layout file
│           ├── for eMMC and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│           ├── FlashLayout_emmc_stm32mp157c-ev1-trusted.tsv       Flash layout file
│           ├── for eMMC and trusted boot chain → STM32MP157C-EV1
│           ├── FlashLayout_nand-4-256_stm32mp157c-ev1-optee.tsv   Flash layout file
│           ├── for NAND Flash and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│           ├── FlashLayout_nand-4-256_stm32mp157c-ev1-trusted.tsv Flash layout file
│           ├── for NAND Flash and trusted boot chain → STM32MP157C-EV1
│           ├── FlashLayout_nor-emmc_stm32mp157c-ev1-optee.tsv     Flash layout file
│           ├── for NOR Flash (and eMMC) and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│           ├── FlashLayout_nor-emmc_stm32mp157c-ev1-trusted.tsv   Flash layout file
│           ├── for NOR Flash (and eMMC) and trusted boot chain → STM32MP157C-EV1
│           ├── FlashLayout_nor-nand-4-256_stm32mp157c-ev1-optee.tsv Flash layout file
│           ├── for NOR Flash (and NAND Flash) and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│           ├── FlashLayout_nor-nand-4-256_stm32mp157c-ev1-trusted.tsv Flash layout file
│           ├── for NOR Flash (and NAND Flash) and trusted boot chain → STM32MP157C-EV1
│           ├── FlashLayout_nor-sdcard_stm32mp157c-ev1-optee.tsv   Flash layout file
│           ├── for NOR Flash (and microSD card) and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│           ├── FlashLayout_nor-sdcard_stm32mp157c-ev1-trusted.tsv Flash layout file
│           ├── for NOR Flash (and microSD card) and trusted boot chain → STM32MP157C-EV1
│           ├── FlashLayout_sdcard_stm32mp157a-dk1-basic.tsv       Flash layout file
│           ├── for microSD card and basic boot chain → STM32MP157A-DK1
│           ├── FlashLayout_sdcard_stm32mp157a-dk1-optee.tsv       Flash layout file
│           ├── for microSD card and trusted boot chain, with OP-TEE OS → STM32MP157A-DK1
│           ├── FlashLayout_sdcard_stm32mp157a-dk1-trusted.tsv     Flash layout file
│           ├── for microSD card and trusted boot chain (recommended setup) → STM32MP157A-DK1
│           ├── FlashLayout_sdcard_stm32mp157c-dk2-basic.tsv       Flash layout file
│           ├── for microSD card and basic boot chain → STM32MP157C-DK2
│           ├── FlashLayout_sdcard_stm32mp157c-dk2-optee.tsv       Flash layout file
│           ├── for microSD card and trusted boot chain, with OP-TEE OS → STM32MP157C-DK2
│           ├── FlashLayout_sdcard_stm32mp157c-dk2-trusted.tsv     Flash layout file
│           ├── for microSD card and trusted boot chain (recommended setup) → STM32MP157C-DK2
│           ├── FlashLayout_sdcard_stm32mp157c-ev1-basic.tsv       Flash layout file
│           ├── for microSD card and basic boot chain → STM32MP157C-EV1
│           └── FlashLayout_sdcard_stm32mp157c-ev1-optee.tsv       Flash layout file
```



Example of directory structure for Packages



Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)

Linux[®] is a registered trademark of Linus Torvalds.

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

Trusted Firmware for Arm[®] Cortex[®]-A

Open Portable Trusted Execution Environment

Operating System

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

former spelling for e•MMC ('e' in italic)

Arm[®] is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex[®]

Microprocessor Unit

Board support package

Cortex Microcontroller Software Interface Standard

Hardware Abstraction Layer

Firmware Image Package is a packaging format used by TF-A

Advanced Linux sound architecture



Graphics Processing Units

Open Graphics Library (See <http://www.opengl.org/> for more details)

Open Vector Graphics (See <http://www.khronos.org/opencv/> for more details)

Khronos Native Platform Graphics Interface (See <http://www.khronos.org/egl/> for more details)

First Stage Boot Loader

Second Stage Boot Loader

Flash memory shortened to gain space in titles, tables and block diagrams