



Example of directory structure for Packages



Example of directory structure for Packages

Stable: 24.06.2020 - 13:13 / Revision: 23.06.2020 - 07:31

Contents

1 Article purpose	2
2 Creating the structure	2
3 Focus on the Starter Package directory	3
4 Focus on the Developer Package directory	5
5 Focus on the Distribution Package directory	8
6 Appendix A: directory structure after build (Developer Package)	9
7 Appendix B: directory structure after build (Distribution Package)	11

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-v1.2.0** 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):



Example of directory structure for Packages

```
PC $> mkdir STM32MP15-Ecosystem-v1.2.0
PC $> cd STM32MP15-Ecosystem-v1.2.0
```

- 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-v1.2.0  STM32MPU Embedded Software release
├── Developer-Package       Developer Package installation directory
├── Distribution-Package    Distribution Package installation directory
└── Starter-Package        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-v1.2.0  STM32MPU Embedded Software release
├── Developer-Package       Developer Package installation
│                           directory
│   ├── SDK                SDK for OpenSTLinux distribution
│   ├── STM32Cube_FW_MP1_V1.2.0  STM32CubeMP1 Package
│   └── stm32mp1-openstlinux-20-02-19  Linux kernel, U-Boot, TF-A and OP-TEE OS source
│                           code (OpenSTLinux distribution)
├── Distribution-Package    Distribution Package installation
│                           directory
│   └── stm32mp1-openstlinux-20-02-19  OpenSTLinux distribution (full source code and
│                           OpenEmbedded-based build framework)
├── Starter-Package        Starter Package installation
│                           directory
│   └── stm32mp1-openstlinux-20-02-19  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 chains overview](#) article for details.

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



Example of directory structure for Packages

```
Starter-Package
├── stm32mp1-openstlinux-20-02-19
│   └── 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_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_stm32mp157a-dk1-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) → 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-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
│               ├── 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 for microSD card and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│               ├── FlashLayout_sdcard_stm32mp157c-ev1-trusted.tsv      Flash
│               ├── layout file for microSD card and trusted boot chain (recommended setup) → STM32MP157C-
│               │ EV1
│               ├── scripts
│               │ └── create_sdcard_from_flashlayout.sh
│               ├── st-image-bootfs-openstlinux-weston-stm32mp1.ext4    Binary for
│               └── bootfs partition
│                   ├── st-image-bootfs-openstlinux-weston-stm32mp1.manifest
│                   ├── st-image-userfs-openstlinux-weston-stm32mp1.ext4 Binary for
│                   └── userfs partition
│                       ├── st-image-userfs-openstlinux-weston-stm32mp1.manifest
│                       └── st-image-vendorfs-openstlinux-weston-stm32mp1.ext4 Binary for
```



Example of directory structure for Packages

vendorfs partition	st-image-weston-openstlinux-weston-stm32mp1.ext4	Binary for
rootfs partition	st-image-weston-openstlinux-weston-stm32mp1.license	
	st-image-weston-openstlinux-weston-stm32mp1-license_content.html	
	st-image-weston-openstlinux-weston-stm32mp1.manifest	
	st-image-weston-openstlinux-weston-stm32mp1_nand_4_256_multivolume.ubi	
	tee-[header/pageable/pager]_v2-stm32mp157a-dk1-optee.stm32	Binaries
for OP-TEE OS runtime services (trusted boot chain) → STM32MP157A-DK1	tee-[header/pageable/pager]_v2-stm32mp157c-dk2-optee.stm32	Binaries
for OP-TEE OS runtime services (trusted boot chain) → STM32MP157C-DK2	tee-[header/pageable/pager]_v2-stm32mp157c-ev1-optee.stm32	Binaries
for OP-TEE OS runtime services (trusted boot chain) → STM32MP157C-EV1	tf-a-stm32mp157a-dk1-optee.stm32	TF-A
binary for FSBL partition (trusted boot chain), with OP-TEE OS → STM32MP157A-DK1	tf-a-stm32mp157a-dk1-trusted.stm32	TF-A
binary for FSBL partition (trusted boot chain) → STM32MP157A-DK1	tf-a-stm32mp157c-dk2-optee.stm32	TF-A
binary for FSBL partition (trusted boot chain), with OP-TEE OS → STM32MP157C-DK2	tf-a-stm32mp157c-dk2-trusted.stm32	TF-A
binary for FSBL partition (trusted boot chain) → STM32MP157C-DK2	tf-a-stm32mp157c-ev1-optee.stm32	TF-A
binary for FSBL partition (trusted boot chain), with OP-TEE OS → STM32MP157C-EV1	tf-a-stm32mp157c-ev1-trusted.stm32	TF-A
binary for FSBL partition (trusted boot chain) → STM32MP157C-EV1	u-boot-spl.stm32-stm32mp157a-dk1-basic	U-Boot
binary for FSBL partition (basic boot chain) → STM32MP157A-DK1	u-boot-spl.stm32-stm32mp157c-dk2-basic	U-Boot
binary for FSBL partition (basic boot chain) → STM32MP157C-DK2	u-boot-spl.stm32-stm32mp157c-ev1-basic	U-Boot
binary for FSBL partition (basic boot chain) → STM32MP157C-EV1	u-boot-stm32mp157a-dk1-basic.img	U-Boot
binary for SSBL partition (basic boot chain) → STM32MP157A-DK1	u-boot-stm32mp157a-dk1-optee.stm32	U-Boot
binary for SSBL partition (trusted boot chain), with OP-TEE OS → STM32MP157A-DK1	u-boot-stm32mp157a-dk1-trusted.stm32	U-Boot
binary for SSBL partition (trusted boot chain) → STM32MP157A-DK1	u-boot-stm32mp157c-dk2-basic.img	U-Boot
binary for SSBL partition (basic boot chain) → STM32MP157C-DK2	u-boot-stm32mp157c-dk2-optee.stm32	U-Boot
binary for SSBL partition (trusted boot chain), with OP-TEE OS → STM32MP157C-DK2	u-boot-stm32mp157c-dk2-trusted.stm32	U-Boot
binary for SSBL partition (trusted boot chain) → STM32MP157C-DK2	u-boot-stm32mp157c-ev1-basic.img	U-Boot
binary for SSBL partition (basic boot chain) → STM32MP157C-EV1	u-boot-stm32mp157c-ev1-optee.stm32	U-Boot
binary for SSBL partition (trusted boot chain), with OP-TEE OS → STM32MP157C-EV1	u-boot-stm32mp157c-ev1-trusted.stm32	U-Boot
binary for SSBL partition (trusted boot chain) → STM32MP157C-EV1		

4 Focus on the Developer Package directory

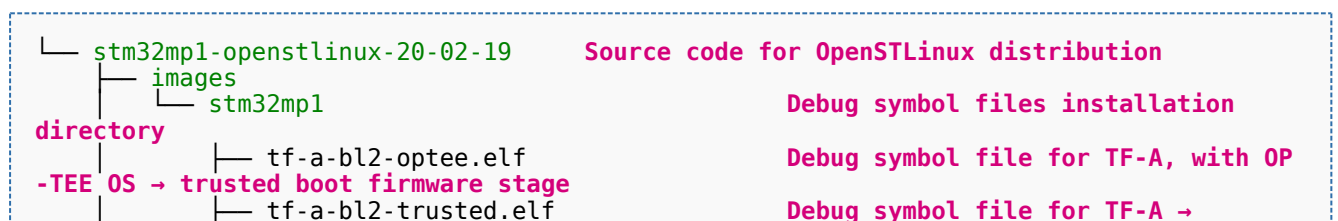
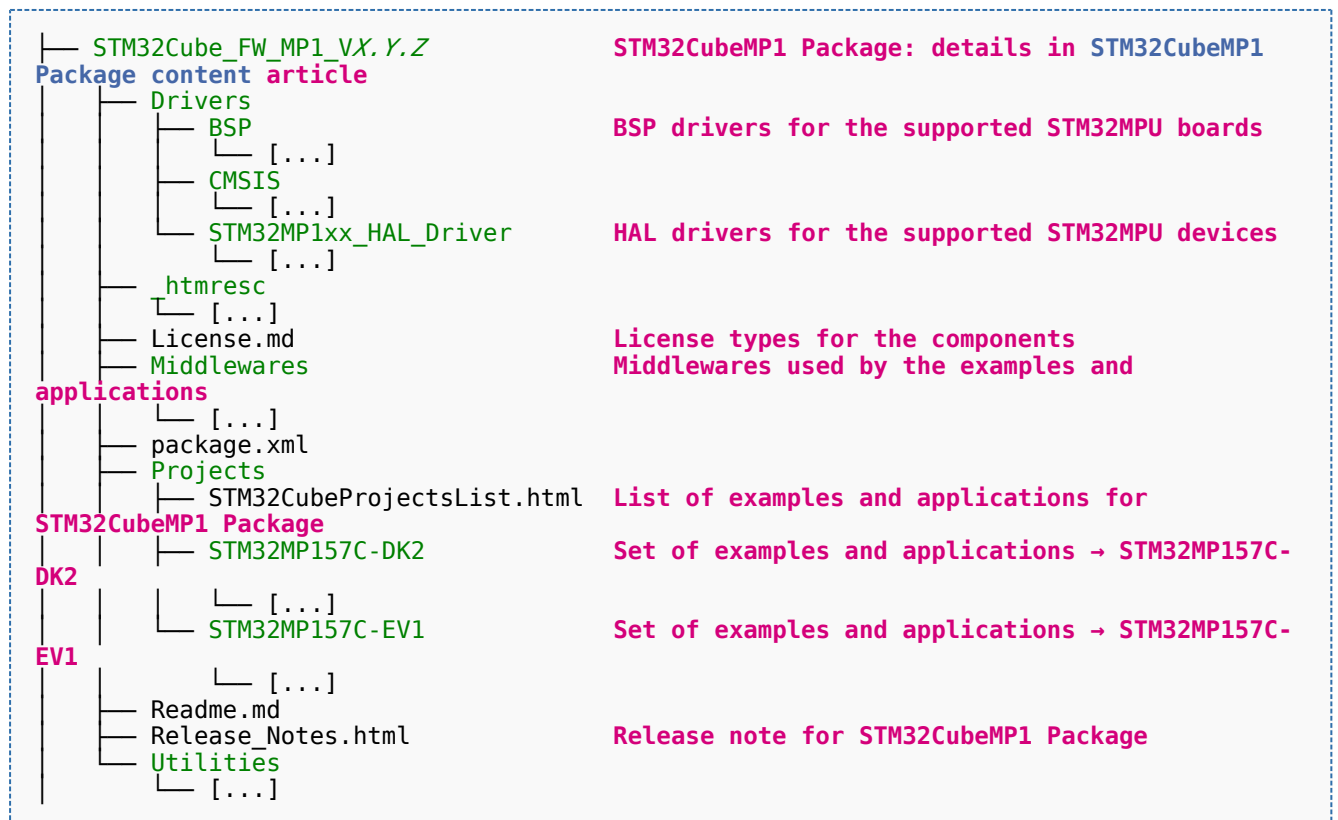
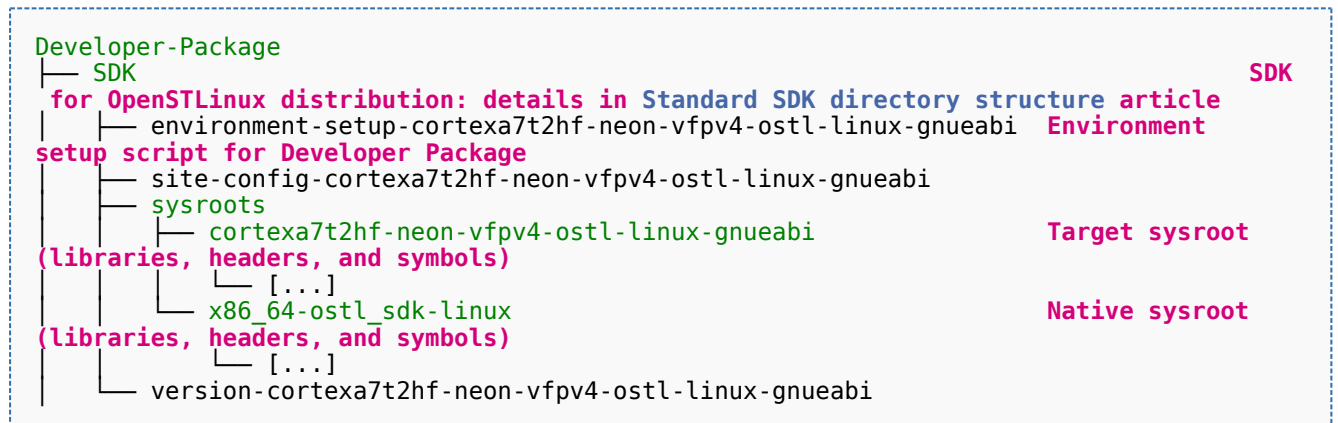
The *Developer-Package* directory contains:

- The source code for the following OpenSTLinux software packages (development for Arm[®] Cortex[®]-A processor):
 - Linux[®] kernel
 - U-Boot



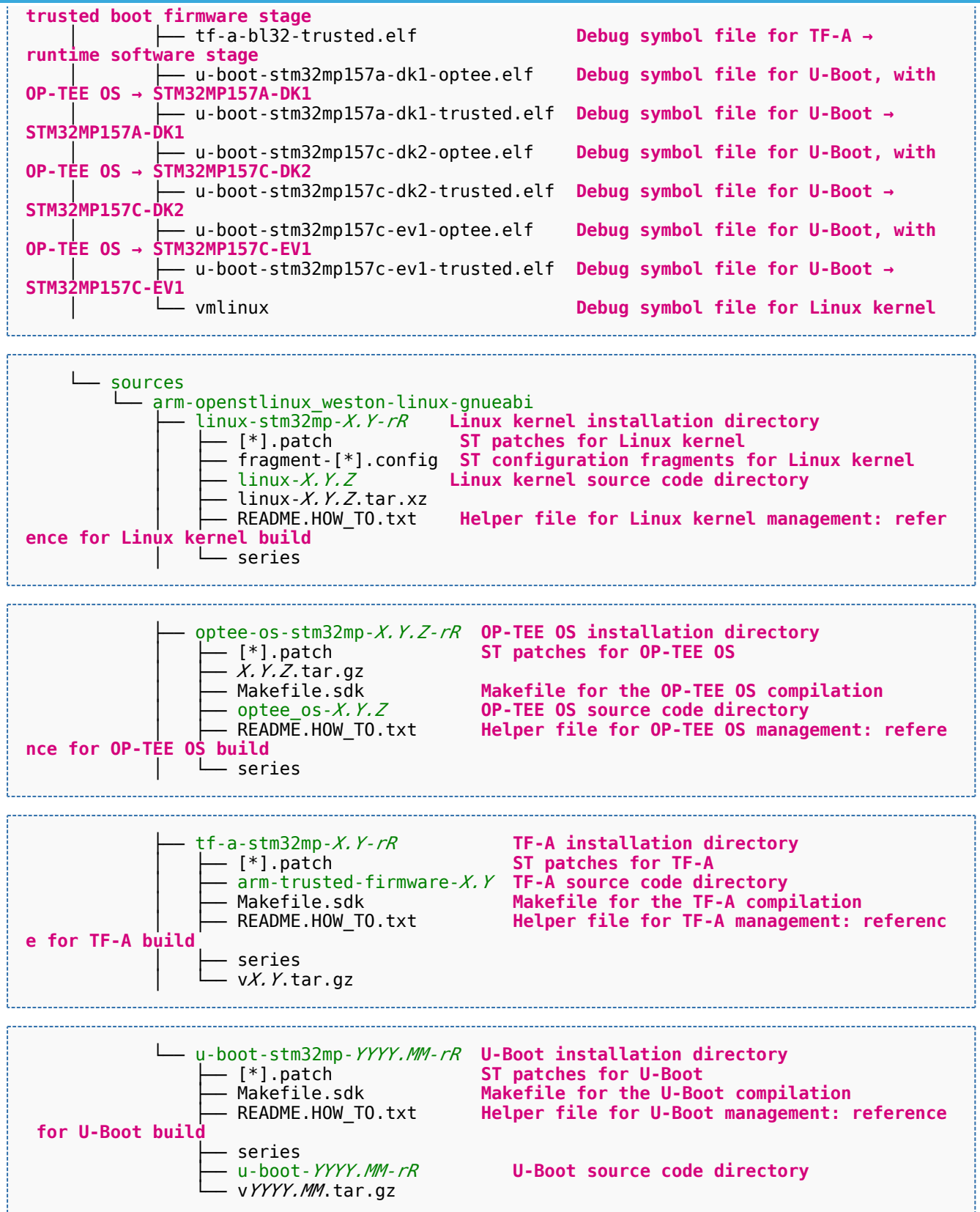
Example of directory structure for Packages

- 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

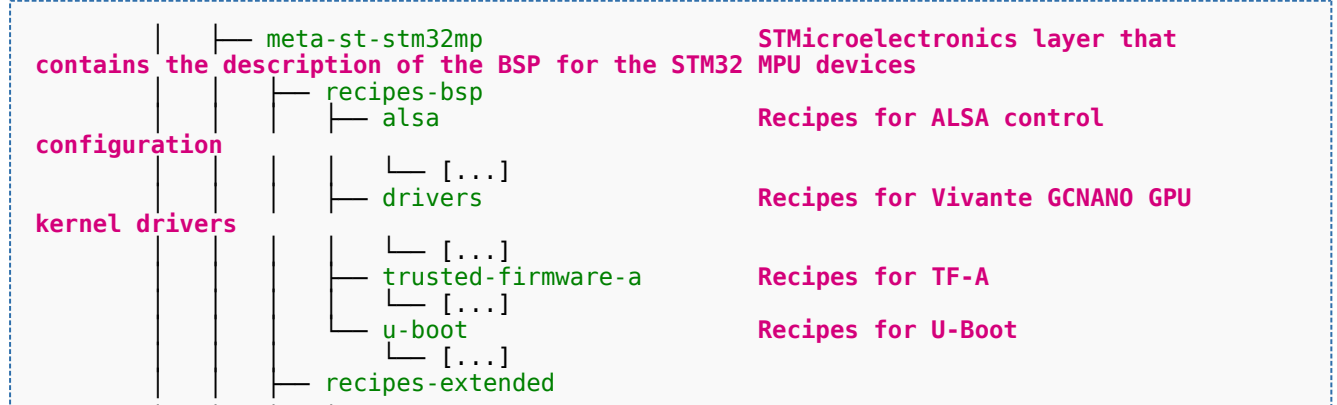
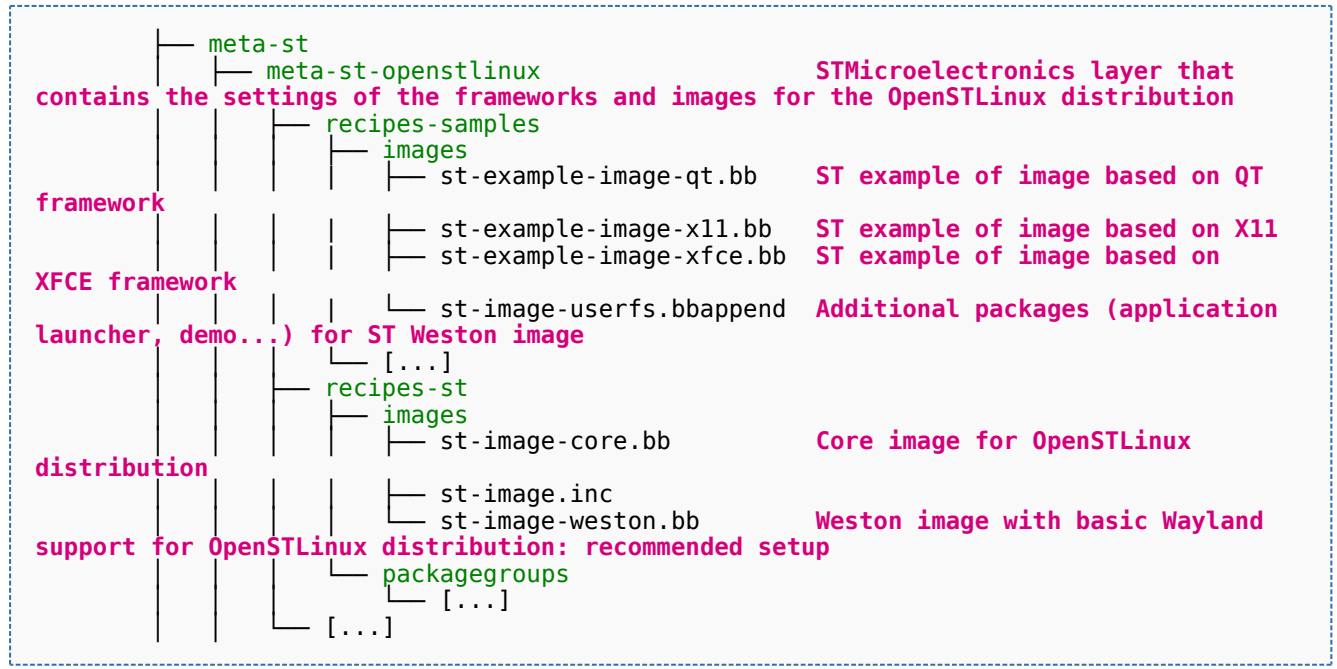
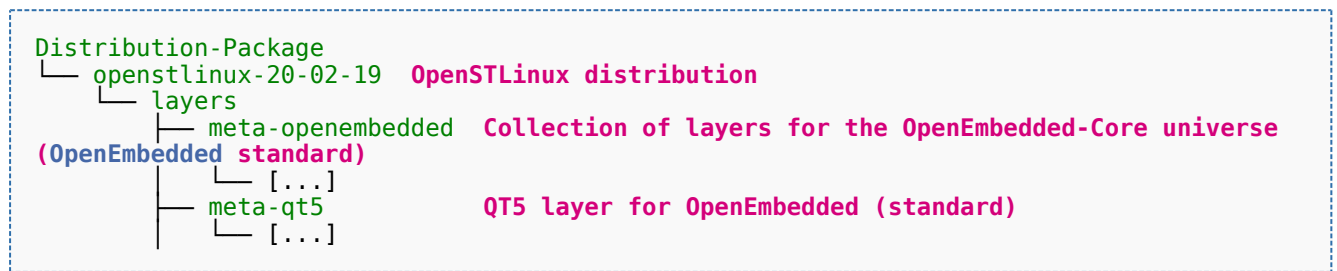


Appendix A shows the structure of the Linux kernel, U-Boot, TF-A and OP-TEE OS installation directories after these software packages have been built.



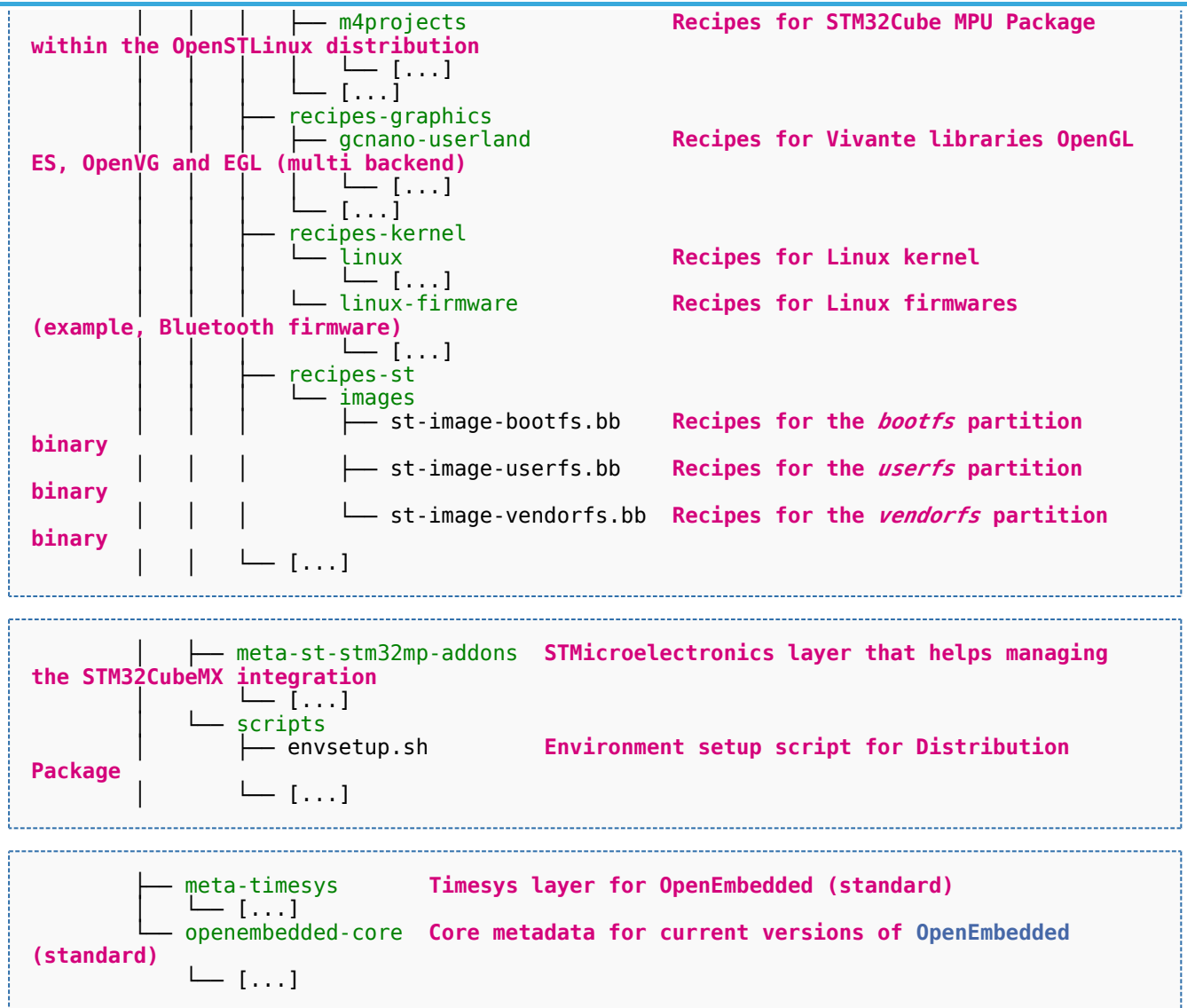
5 Focus on the Distribution Package directory

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.





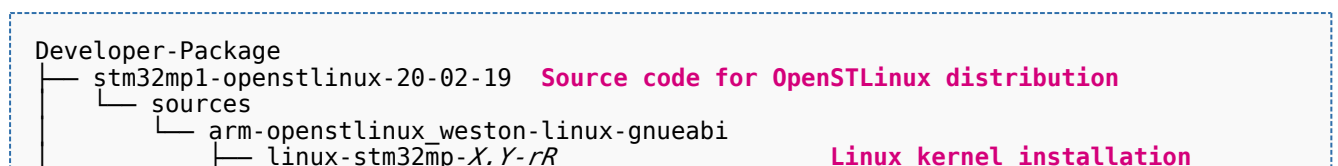
Example of directory structure for Packages



Appendix B shows the structure of the build directory.

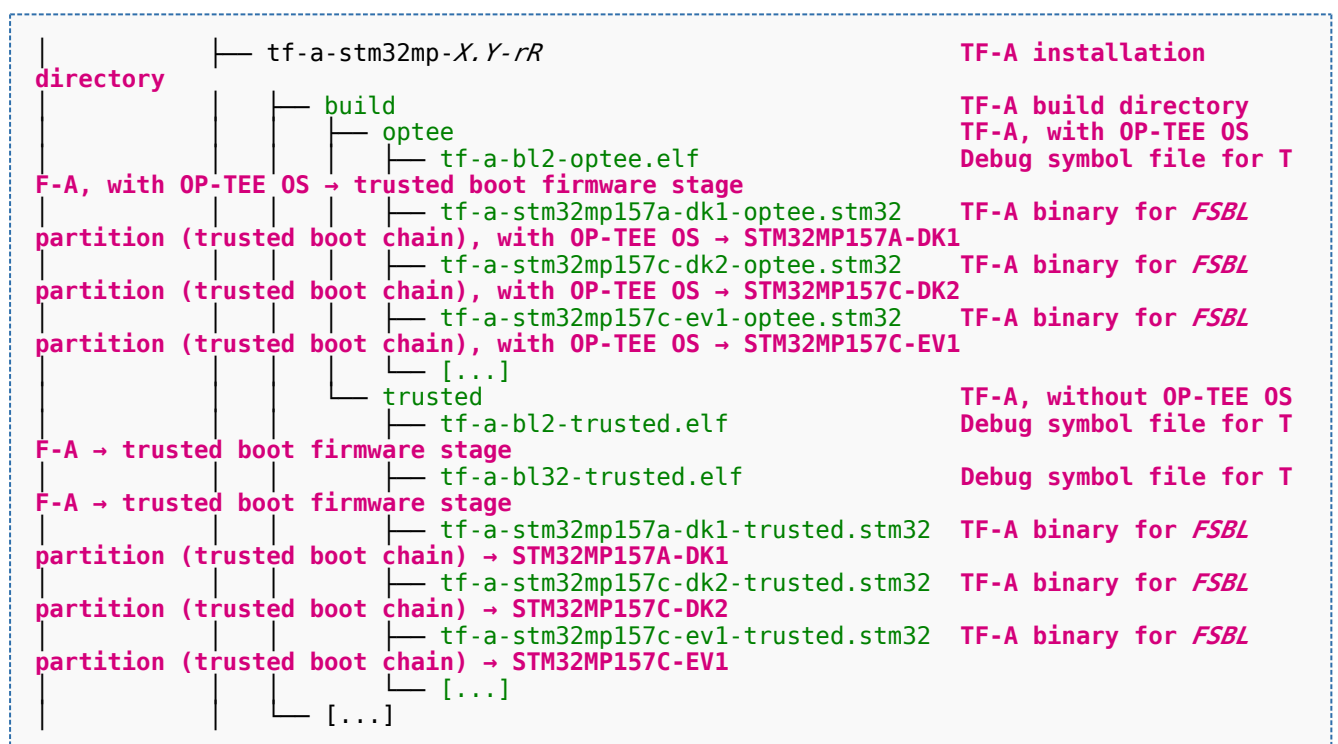
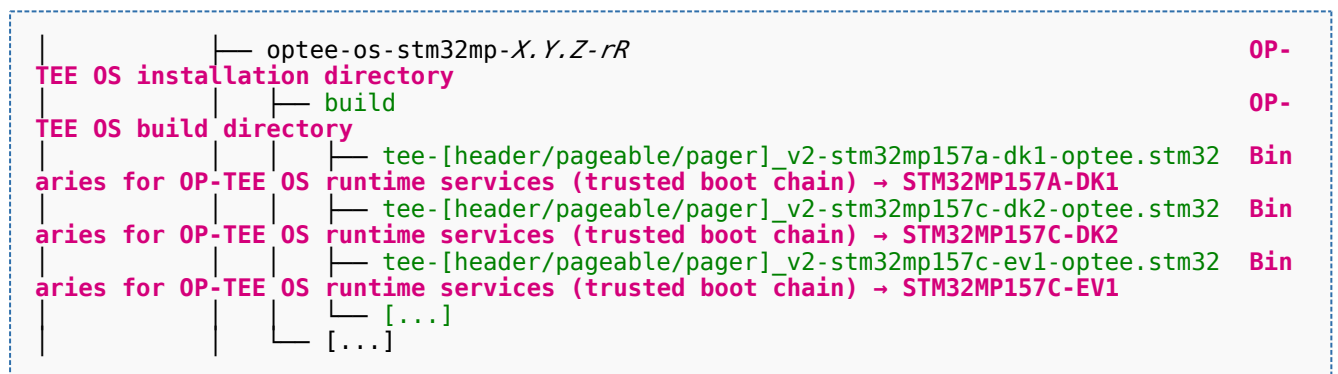
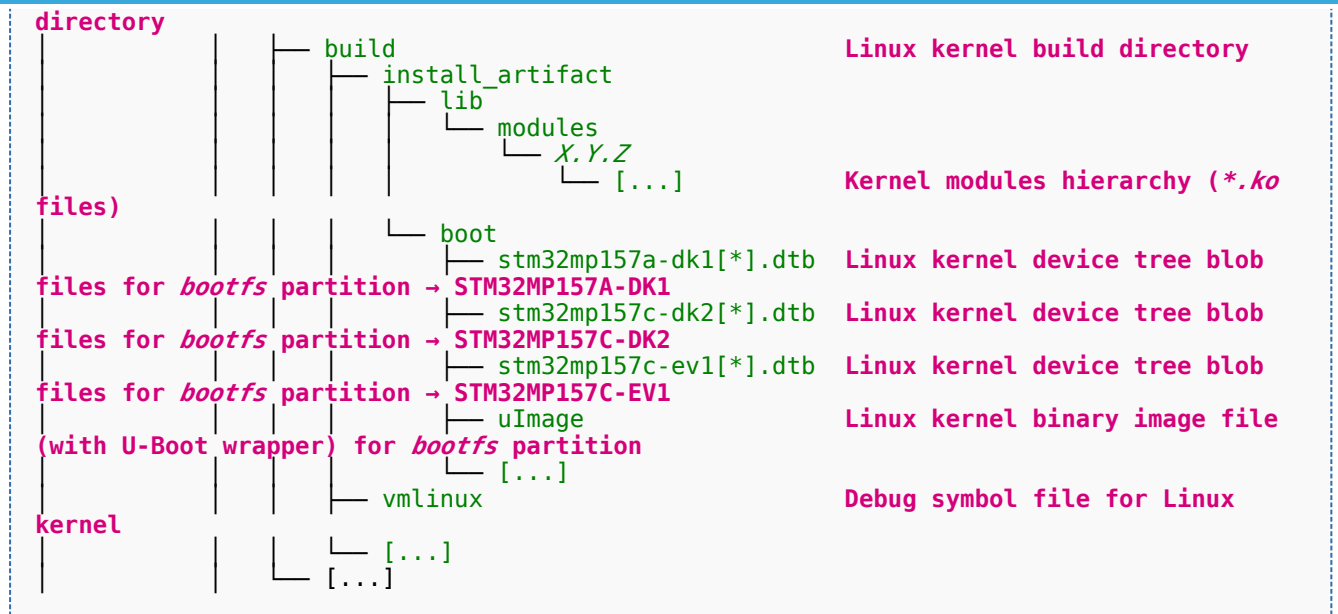
6 Appendix A: directory structure after build (Developer Package)

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.



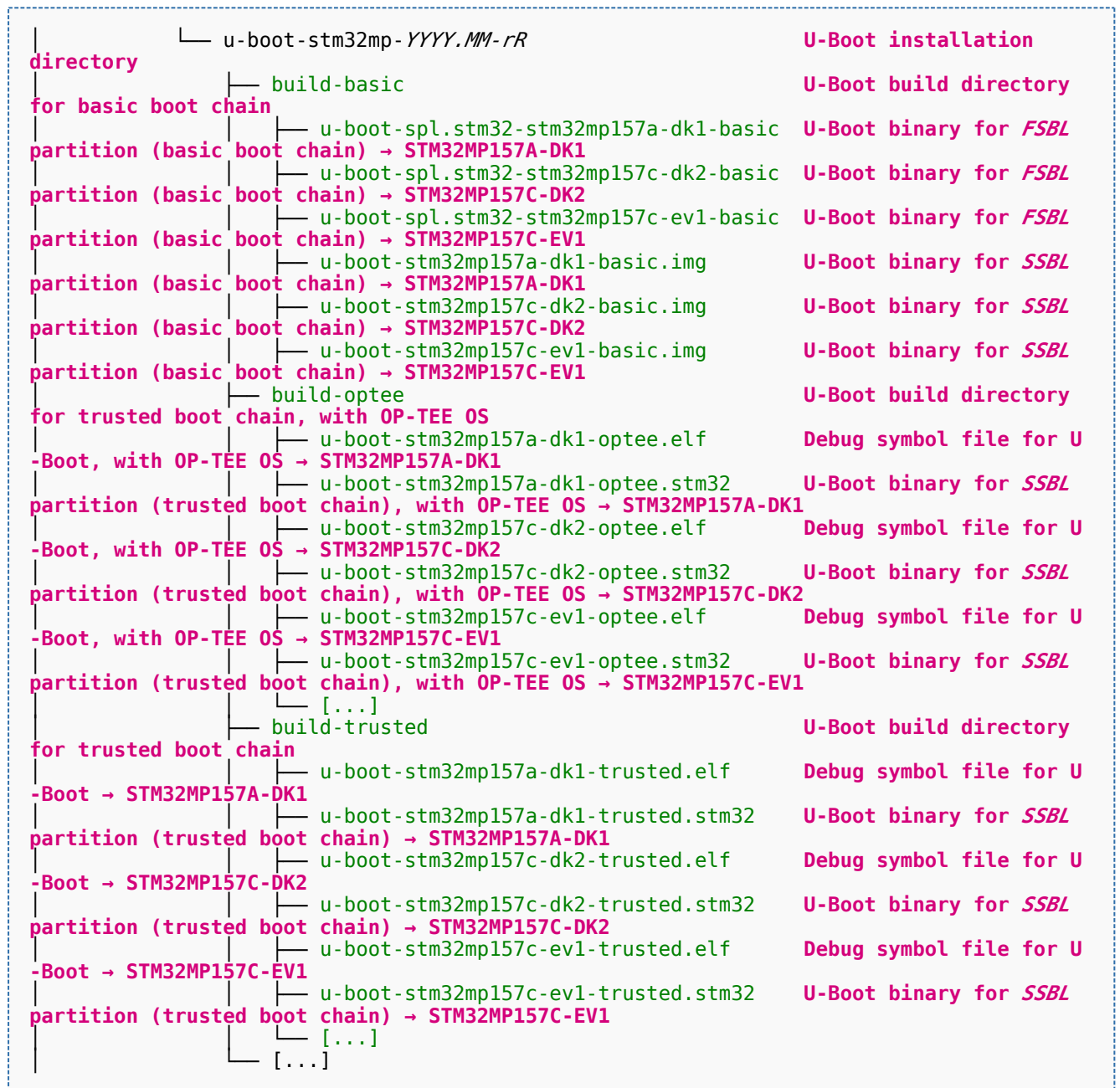


Example of directory structure for Packages





Example of directory structure for Packages



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.



Example of directory structure for Packages

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-20-02-19 /build-openstlinuxweston-stm32mp/tmp-glibc
/deplo
├── 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 for microSD card and trusted boot chain, with OP-TEE OS → STM32MP157C-EV1
│           ├── FlashLayout_sdcard_stm32mp157c-ev1-trusted.tsv    Flash layout
│           ├── file for microSD card and trusted boot chain (recommended setup) → STM32MP157C-EV1
│           └── [...]
│       └── scripts
│           └── create_sdcard_from_flashlayout.sh
```



Example of directory structure for Packages

fs partition	st-image-bootfs-openstlinux-weston-stm32mp1.ext4	Binary for <i>boot</i>
fs partition	st-image-userfs-openstlinux-weston-stm32mp1.ext4	Binary for <i>user</i>
orfs partition	st-image-vendorfs-openstlinux-weston-stm32mp1.ext4	Binary for <i>vend</i>
fs partition	st-image-weston-openstlinux-weston-stm32mp1.ext4	Binary for <i>root</i>
device tree blob files for <i>bootfs</i> partition → STM32MP157A-DK1	stm32mp157a-dk1[*].dtb	Linux kernel
device tree blob files for <i>bootfs</i> partition → STM32MP157C-DK2	stm32mp157c-dk2[*].dtb	Linux kernel
device tree blob files for <i>bootfs</i> partition → STM32MP157C-EV1	stm32mp157c-e[*].dtb	Linux kernel
-TEE OS runtime services (trusted boot chain) → STM32MP157A-DK1	tee-[header/pageable/pager]_v2-stm32mp157a-dk1-optee.stm32	Binaries for OP
-TEE OS runtime services (trusted boot chain) → STM32MP157C-DK2	tee-[header/pageable/pager]_v2-stm32mp157c-dk2-optee.stm32	Binaries for OP
-TEE OS runtime services (trusted boot chain) → STM32MP157C-EV1	tee-[header/pageable/pager]_v2-stm32mp157c-ev1-optee.stm32	Binaries for OP
file for TF-A, with OP-TEE OS → trusted boot firmware stage	tf-a-bl2-optee.elf	Debug symbol
file for TF-A → trusted boot firmware stage	tf-a-bl2-trusted.elf	Debug symbol
file for TF-A → runtime software stage	tf-a-bl32-trusted.elf	Debug symbol
for <i>FSBL</i> partition (trusted boot chain), with OP-TEE OS → STM32MP157A-DK1	tf-a-stm32mp157a-dk1-optee.stm32	TF-A binary
for <i>FSBL</i> partition (trusted boot chain) → STM32MP157A-DK1	tf-a-stm32mp157a-dk1-trusted.stm32	TF-A binary
for <i>FSBL</i> partition (trusted boot chain), with OP-TEE OS → STM32MP157C-DK2	tf-a-stm32mp157c-dk2-optee.stm32	TF-A binary
for <i>FSBL</i> partition (trusted boot chain) → STM32MP157C-DK2	tf-a-stm32mp157c-dk2-trusted.stm32	TF-A binary
for <i>FSBL</i> partition (trusted boot chain), with OP-TEE OS → STM32MP157C-EV1	tf-a-stm32mp157c-ev1-optee.stm32	TF-A binary
for <i>FSBL</i> partition (trusted boot chain) → STM32MP157C-EV1	tf-a-stm32mp157c-ev1-trusted.stm32	TF-A binary
for <i>FSBL</i> partition (basic boot chain) → STM32MP157A-DK1	u-boot-spl.stm32-stm32mp157a-dk1-basic	U-Boot binary
for <i>FSBL</i> partition (basic boot chain) → STM32MP157C-DK2	u-boot-spl.stm32-stm32mp157c-dk2-basic	U-Boot binary
for <i>FSBL</i> partition (basic boot chain) → STM32MP157C-EV1	u-boot-spl.stm32-stm32mp157c-ev1-basic	U-Boot binary
for <i>SSBL</i> partition (basic boot chain) → STM32MP157A-DK1	u-boot-stm32mp157a-dk1-basic.img	U-Boot binary
file for U-Boot, with OP-TEE OS → STM32MP157A-DK1	u-boot-stm32mp157a-dk1-optee.elf	Debug symbol
for <i>SSBL</i> partition (trusted boot chain), with OP-TEE OS → STM32MP157A-DK1	u-boot-stm32mp157a-dk1-trusted.stm32	U-Boot binary
file for U-Boot → STM32MP157A-DK1	u-boot-stm32mp157a-dk1-trusted.stm32	U-Boot binary
for <i>SSBL</i> partition (basic boot chain) → STM32MP157C-DK2	u-boot-stm32mp157c-dk2-basic.img	U-Boot binary
file for U-Boot, with OP-TEE OS → STM32MP157C-DK2	u-boot-stm32mp157c-dk2-optee.elf	Debug symbol
for <i>SSBL</i> partition (trusted boot chain), with OP-TEE OS → STM32MP157C-DK2	u-boot-stm32mp157c-dk2-trusted.stm32	U-Boot binary
file for U-Boot → STM32MP157C-DK2	u-boot-stm32mp157c-dk2-trusted.stm32	U-Boot binary
for <i>SSBL</i> partition (trusted boot chain) → STM32MP157C-DK2	u-boot-stm32mp157c-ev1-basic.img	U-Boot binary



Example of directory structure for Packages

```
for SSBL partition (basic boot chain) → STM32MP157C-EV1
├── 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 SSBL partition (trusted boot chain), with OP-TEE OS → STM32MP157C-EV1
├── u-boot-stm32mp157c-ev1-trusted.elf       Debug symbol
file for U-Boot → STM32MP157C-EV1
├── u-boot-stm32mp157c-ev1-trusted.stm32     U-Boot binary
for SSBL partition (trusted boot chain) → STM32MP157C-EV1
├── uImage                                   Linux kernel
binary image file (with U-Boot wrapper) for bootfs partition
├── vmlinux                                  Debug symbol
file for Linux kernel
├── [...]
└── [...]
```

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

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 eMMC ('e' in *italic*)

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

First Stage Boot Loader

Second Stage Boot Loader

Microprocessor Unit

Board support package

Cortex Microcontroller Software Interface Standard

Hardware Abstraction Layer

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/opengv/> for more details)

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