File:OpenSTLinux OE legend.png

Contents

Stable: 10.11.2020 - 13:25 / Revision: 10.11.2020 - 13:23

- File
- File history
- File usage
- Metadata

No higher resolution available.
OpenSTLinux_OE_legend.png (235 × 73 pixels, file size: 5 KB, MIME type: image/png)
A quality version of this page, approved on *10 November 2020*, was based off this revision.

Source
STMicroelectronics

File history

Click on a date/time to view the file as it appeared at that time.

	Date/Time	Thumbnail	Dimensions	User	Comment
current	13:23, 10 November 2020		235 × 73 (5 KB)	Gerald Baeza (talk | contribs)	
	18:08, 11 November 2017		235 × 73 (2 KB)	Frq08690 (talk | contribs)	

- You cannot overwrite this file.

File usage

The following 4 pages link to this file:

- Android application frameworks overview
- Linux application frameworks overview
- STM32MPU Embedded Software architecture overview
- STM32MPU Embedded Software for Android architecture overview

Metadata

This file contains additional information, probably added from the digital camera or scanner used to create or digitize it. If the file has been modified from its original state, some details may not fully reflect the modified file.

Horizontal resolution	37.48 dpc
Vertical resolution	37.48 dpc
Software used	• www.inkscape.org

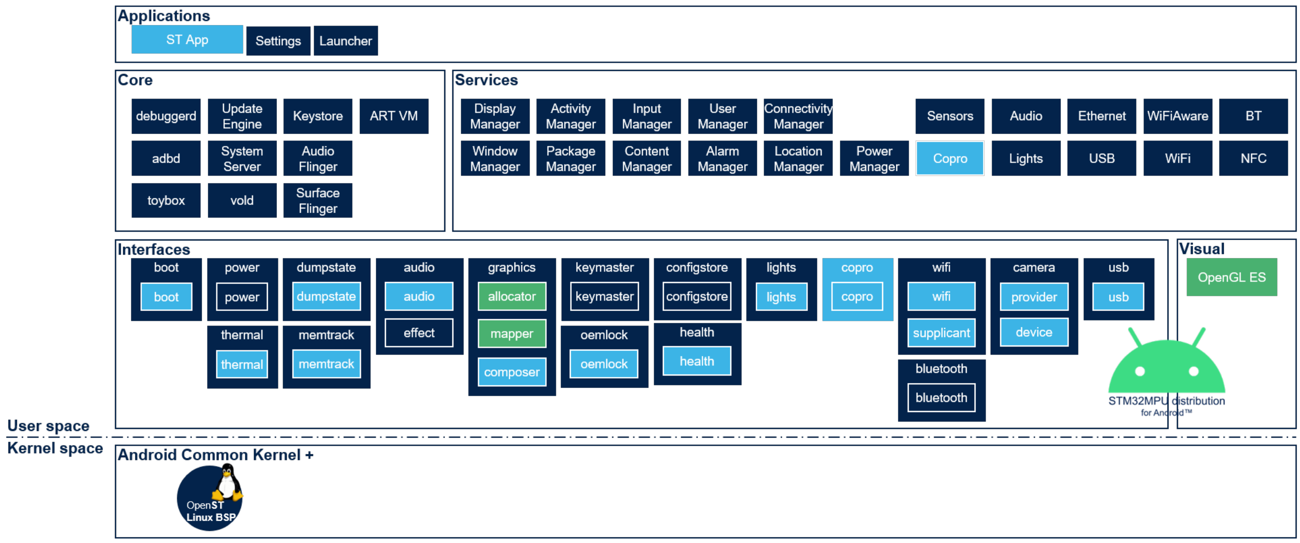
Stable: 09.10.2019 - 12:43 / Revision: 20.08.2019 - 11:12

A quality version of this page, approved on *9 October 2019*, was based off this revision.

The diagram below gives an overview of the Android application frameworks (Linux user space components) that rely on Linux kernel.

This Linux kernel is based on the Android Common Kernel[1] following Google recommendation for its configuration[2].

It shows the main components grouped per functional domain, however it is not exhaustive.



The Android framework is structured in several layers:

- hardware interfaces providing a standard way to configure the underlaying driver, based on the Android HIDL (Hardware Interface Definition Language) mechanism
- the Android core (ART virtual machine, useful daemons including vold, adbd, and debuggerd, and system services)
- Android services (providing interfaces to the application = SDK)
- Android applications (including the launcher)

The Android services are often implemented partly in native cpp, and partly in Java. The JNI (Java Native Interface) IPC mechanism is available to allow communication between the two worlds.

In addition to the standard Android services, a proprietary coprocessor service for Android has been introduced (for development purposes only). The CoproService is composed of two parts:

- firmware management (check running firmware, start/stop firmware, get/set the firmware name)
- TTY management (open/close and read/write the TTY interface). You must implement your own protocol on top of this.

References

- AOSP: https://android.googlesource.com/kernel/common/
- AOSP: https://android.googlesource.com/kernel/configs/

Linux® is a registered trademark of Linus Torvalds.

HAL interface definition language (see https://source.android.com/devices/architecture/hidl)

Android Runtime (see https://source.android.com/devices/tech/dalvik)

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

Java Native Interface (for Android)

Inter-Processor Communication

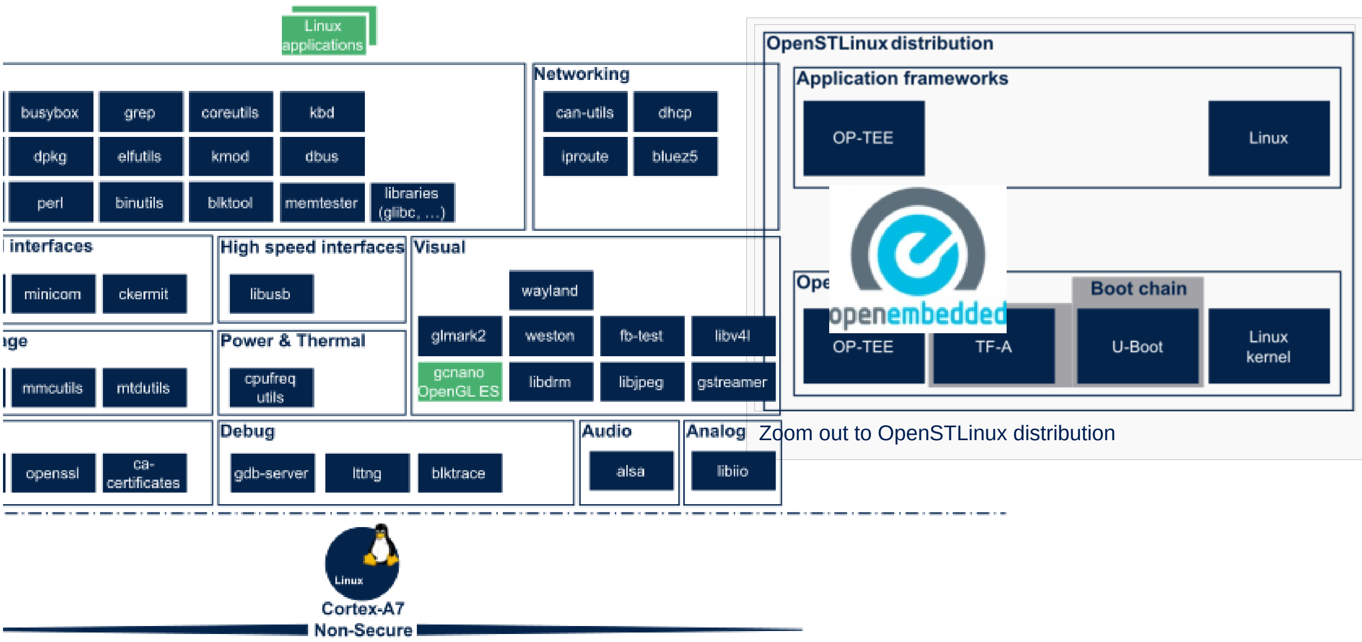
TeleTYpewriter

Android Open Source Project

Stable: 30.01.2020 - 13:51 / Revision: 30.01.2020 - 13:49

A quality version of this page, approved on *30 January 2020*, was based off this revision.

The diagram below gives an overview of the Linux application frameworks (aka Linux user space components), that rely on Linux kernel.
It shows the main components, grouped per functional domains, but it does not intend to be exhaustive: the user can add or remove components via OpenEmbedded recipes to fit with his application needs.



Zoom out to OpenSTLinux distribution

Generally speaking, the application frameworks (aka middlewares) are software pieces that are neither the Linux kernel or the device drivers, nor the applications. In other words, they are the intermediate software between the application software and the kernel or device drivers software. Thus, they are abstraction layers used by different applications to ensure and ease their flexibility, portability, security and interoperability. The main advantage of such application frameworks is to reduce the complexity of the applications with the provided generic services that would be duplicated in the applications otherwise. The counterpart is that they might introduce an overhead, which might impact the overall performance.

Examples of such application frameworks:

- The libusb library for a generic access to USB devices
- The libiio library to ease the development of software interfacing Linux Industrial I/O (IIO) devices
- The Advanced Linux Sound Architecture (ALSA) libraries for audio functionalities
- The Wayland-Weston display/graphic framework
- The GStreamer multimedia framework (library for constructing graphs of media-handling components)
- and much more as illustrated in the figure above

Linux® is a registered trademark of Linus Torvalds.

also known as

Industrial I/O Linux subsystem

Advanced Linux sound architecture

Stable: 25.09.2020 - 09:15 / Revision: 25.09.2020 - 09:13

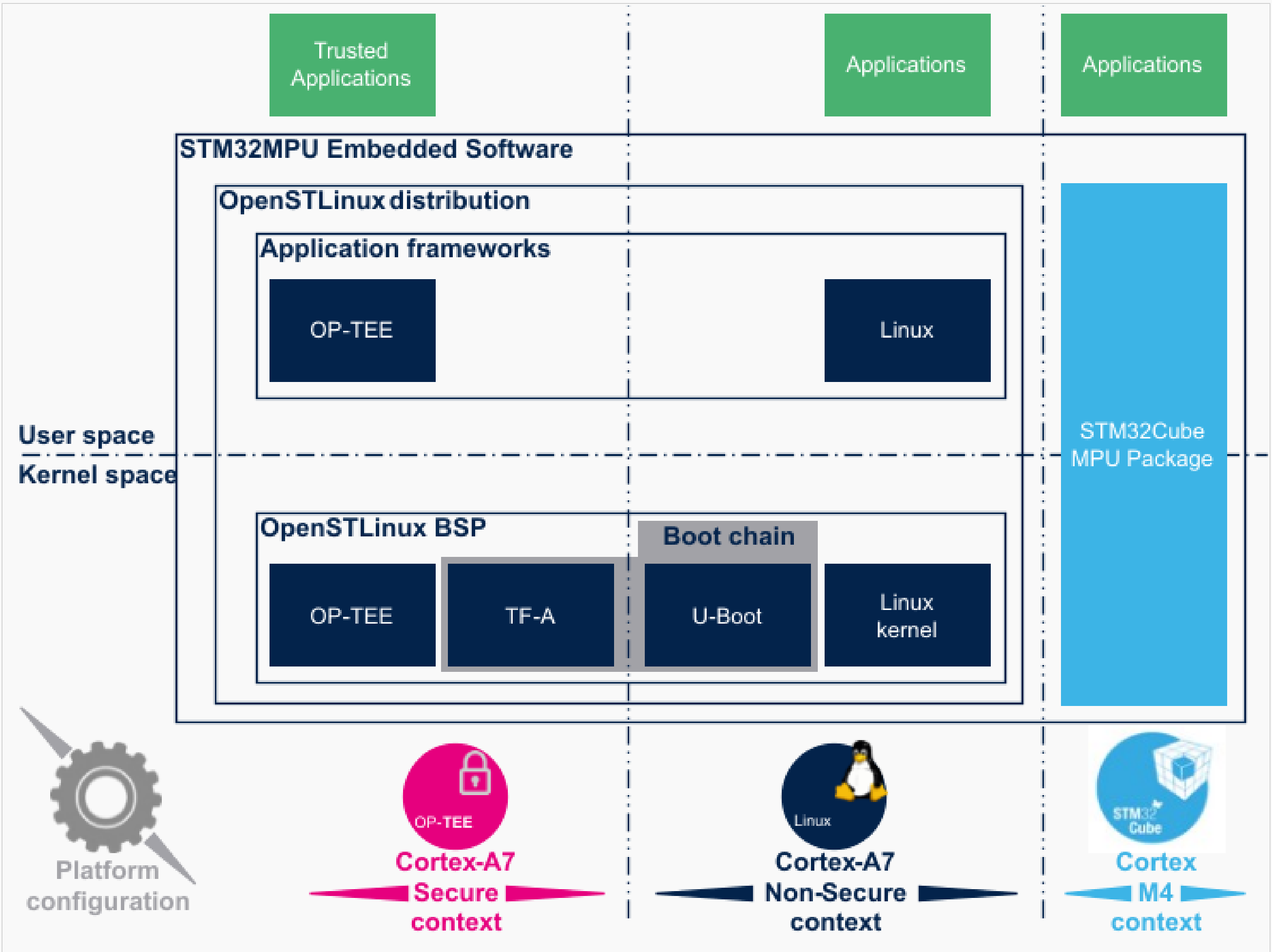
A quality version of this page, approved on *25 September 2020*, was based off this revision.

1 STM32MPU Embedded Software overview

The diagram below shows STM32MPU Embedded Software distribution main components:

- The **OpenSTLinux distribution**, running on the Arm®Cortex®-A, including:
  - The **OpenSTLinux BSP** with:
    - The boot chain based on TF-A and U-Boot.
    - The OP-TEE secure OS running on the Arm®Cortex®-A in secure mode.
    - The **Linux® kernel** running on the Arm®Cortex®-A in non-secure mode.
  - The **application frameworks** are composed of middlewares relying on the BSP and providing API:
    - on the **OP-TEE** side to run **Trusted Applications (TA)** that allow to manipulate secrets (not visible from the Linux and STM32Cube MPU Package)
    - on the **Linux** side to run **Applications** that typically interact with the user via the display, the touchscreen, etc.
- The **STM32Cube MPU Package** is running on the Arm®Cortex®-M: it is based on HAL drivers and middlewares, like other STM32 microcontrollers, completed with coprocessor management.

The figure below is clickable so that the user can directly jump to one of the sub-levels listed above.



STM32MPU Embedded Software architecture overview.

File:OpenSTLinux OE legend.png

2 Open Source Software (OSS) philosophy

The **Open source software** source code is released under a license in which the copyright holder grants users the rights to study, change and distribute the software to anyone and for any purpose[1].
STMicroelectronics maximizes the using of open source software and contributes to those communities. Notice that, due to the software review life cycle, it can take some time before getting all developments accepted in the communities, so STMicroelectronics can also temporarily provide some source code on github[2], until it is merged in the targeted repository.

3 References

- https://en.wikipedia.org/wiki/Open-source_software
- STM32MP1 Distribution Package

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

Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Application programming interface

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

Open Source Software

Stable: 15.02.2021 - 13:09 / Revision: 15.02.2021 - 13:03

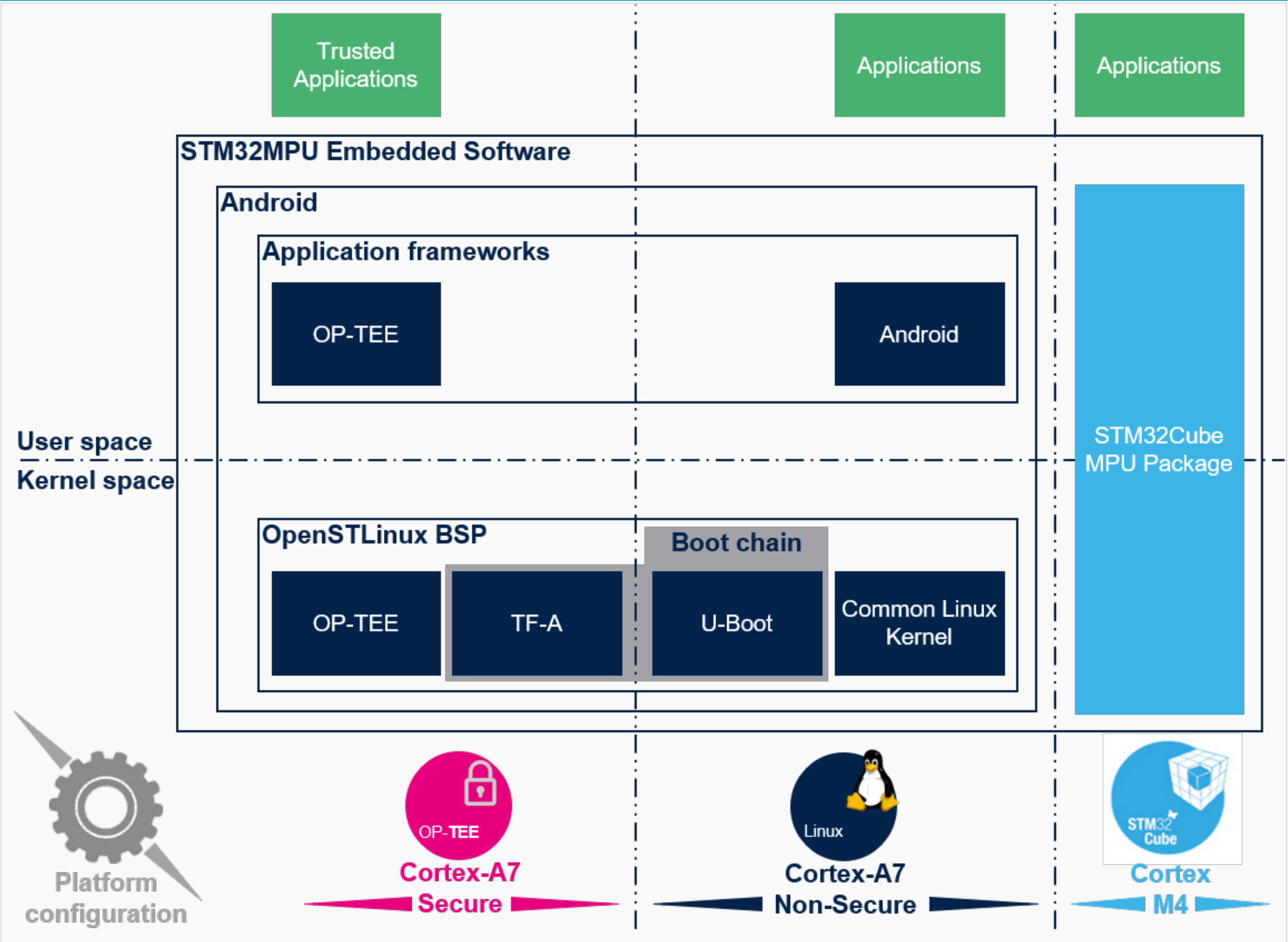
A quality version of this page, approved on *15 February 2021*, was based off this revision.

The diagram below shows STM32MPU Embedded Software distribution for Android main components:

- The **STM32MPU distribution for Android™** running on the Arm®Cortex®-A core. It includes:
  - The **OpenSTLinux BSP** consisting of:
    - The boot chain based on TF-A and U-Boot.
    - The OP-TEE secure OS running on the Arm®Cortex®-A in Secure mode.
    - The **Linux® kernel** running on the Arm®Cortex®-A in Non-secure mode.
  - **Application frameworks** composed of middleware components relying on the BSP and providing a set of APIs:
    - **OP-TEE** APIs to run **Trusted Applications (TA)** that allow manipulating secrets (information not visible from Linux® and from the STM32Cube MPU Package)
    - **Android** APIs to run **Applications** that typically interact with the user via a display or a touchscreen.
- The **STM32Cube MPU Package**, running on the Arm®Cortex®-M. As for STM32 MCUs, it is based on HAL drivers and middleware components and completed with a coprocessor management module.

The figure below provides an overview of the STM32MPU Embedded Software architecture. Click a sublevel block to jump to the corresponding article.

File:OpenSTLinux OE legend.png



STM32MPU Embedded Software architecture overview

1 Open Source Software (OSS) philosophy

The **Open source software** source code is released under a license in which the copyright holder grants users the rights to study, change and distribute the software to anyone and for any purpose[1].
STMicroelectronics maximizes the usage of open source software and contributes to open source software communities. Due to the software review life cycle, it can take some time before getting all developments accepted in the communities, so STMicroelectronics can also temporarily provide some source code on github until it is merged in the targeted repository (see STM32MP1 Distribution Package for Android).

2 References

- https://en.wikipedia.org/wiki/Open-source_software

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

Cortex®

Board support package

Operating System

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Trusted Application

Microprocessor Unit

Hardware Abstraction Layer

Open Source Software