Android application frameworks overview

Contents

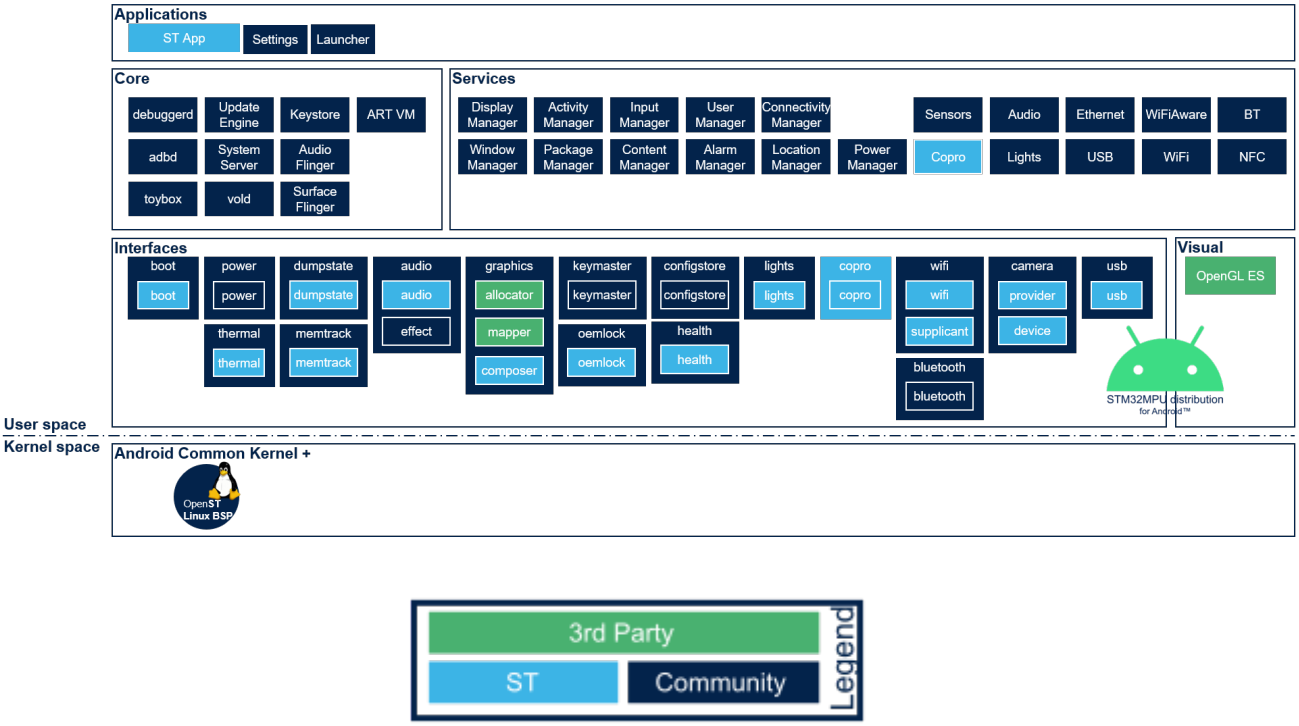
Stable: 09.10.2019 - 12:43 / Revision: 20.08.2019 - 11:12

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/

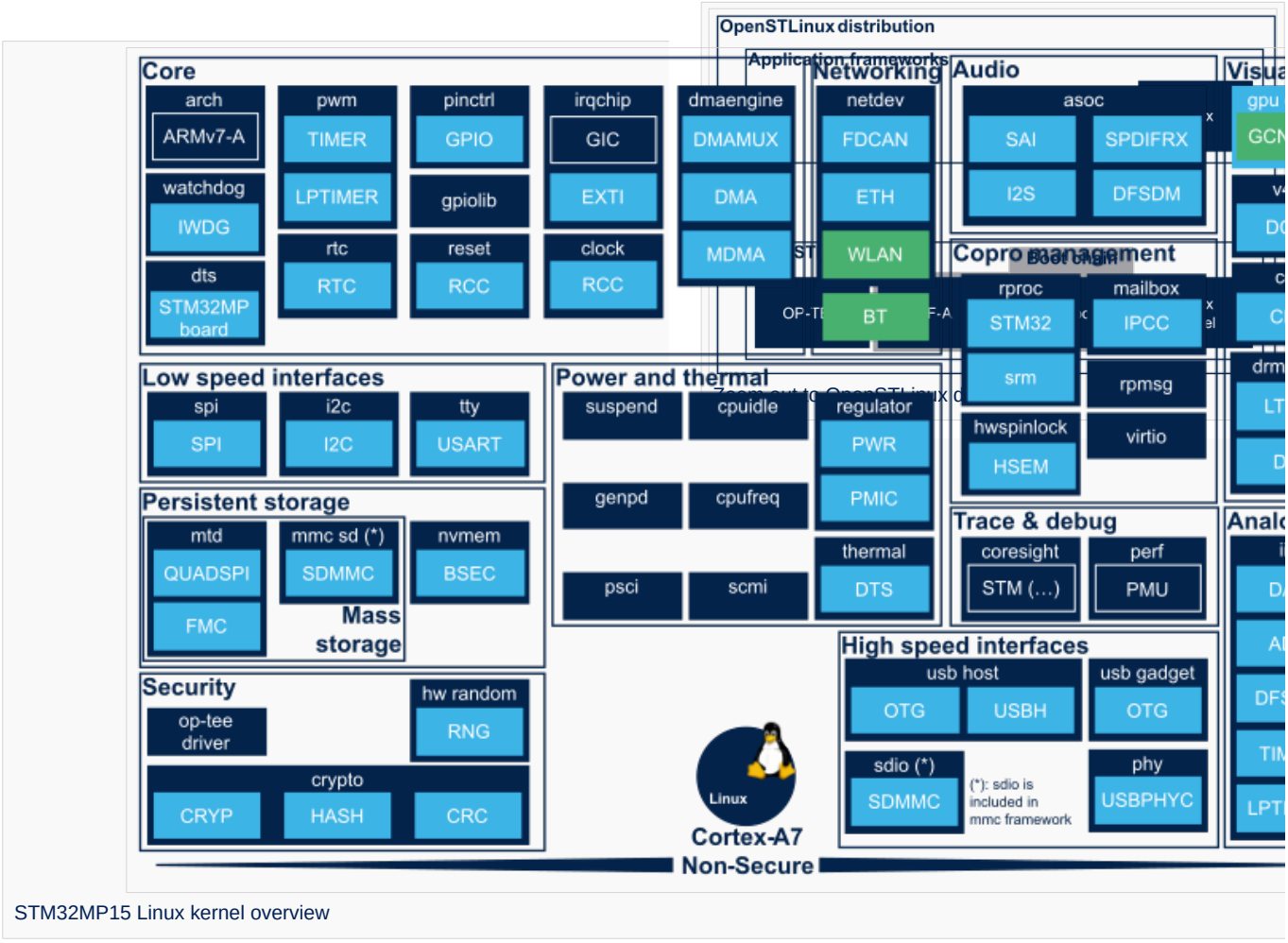
Stable: 05.11.2021 - 08:48 / Revision: 21.10.2021 - 09:59

A quality version of this page, approved on *5 November 2021*, was based off this revision.

This section gives an overview of **Linux® kernel drivers** (UPPERCASE in the figure) implemented for the STM32MP15 support, with their respective **software frameworks** (lowercase in the figure).

The components are grouped per **functional domains**.
Each **Linux framework** is further described in Linux operating system category articles.
Each STM32 MPU **peripheral** is introduced in peripherals overview articles.
Both those sections are reusing the same functional domain split.

The color code, explained in the legend, allows to see the code origin for each component.



STM32MP15 Linux kernel overview