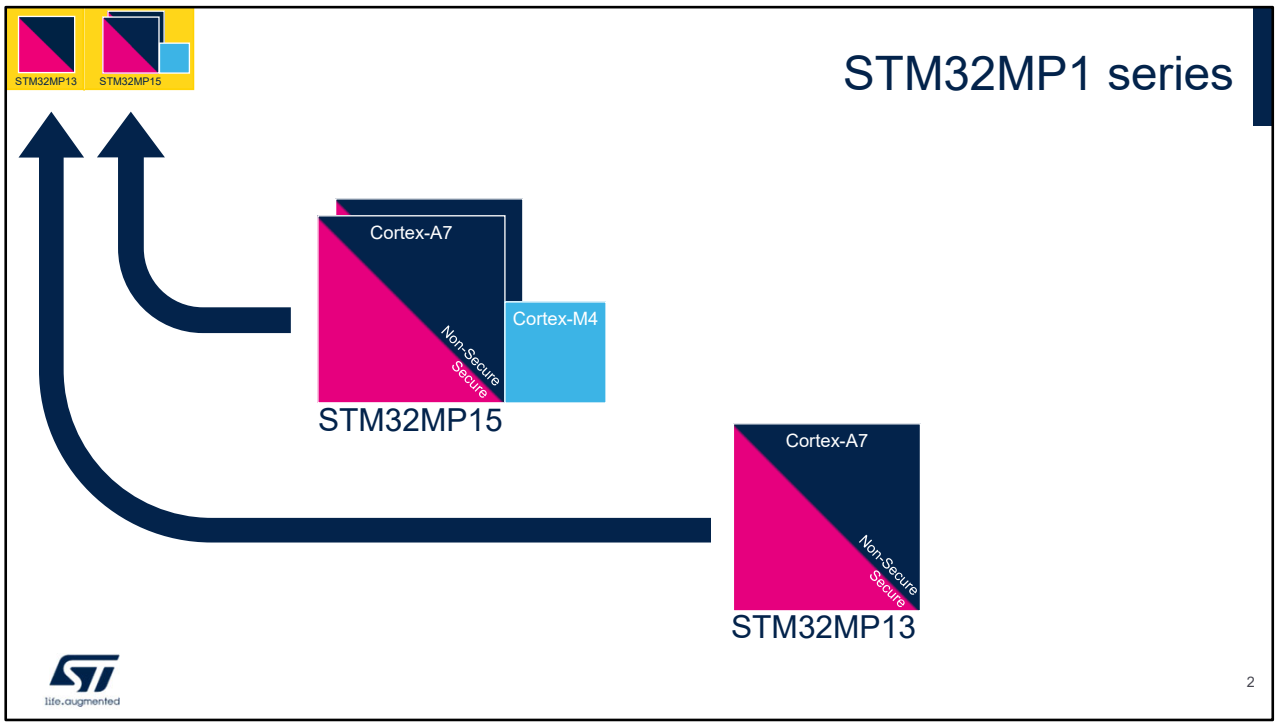
Welcome to this training describing STM32 MPU low power management.

The presentation aims to introduce the STM32 MPU power management mechanisms, to show how they are controlled by OpenSTLinux distribution, and to give pointers to the STM32 MPU Wiki, where you will be able to find up-to-date and more detailed information.

The “STM32MP1 series software architecture and secure boot” online training should be followed before this session, for a proper understanding of the OP-TEE services concept and installation.

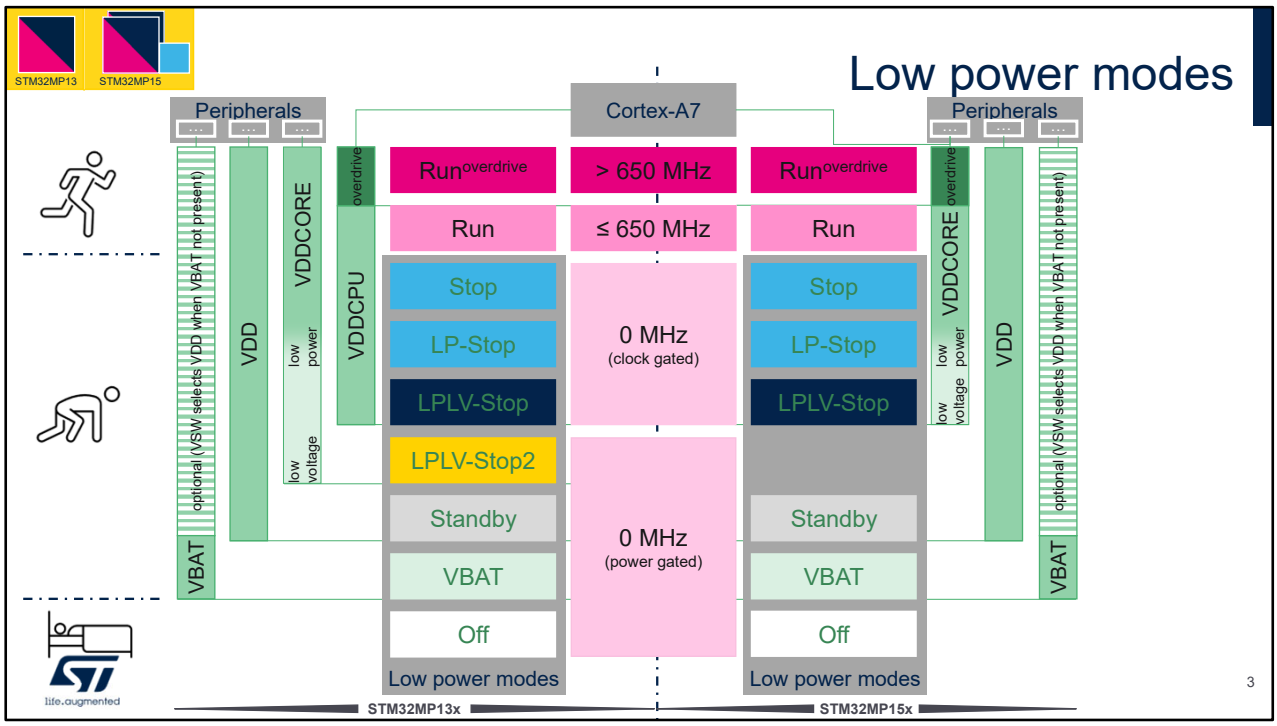


The STM32MP1 microprocessor series is built around Arm Cortex-A7 and Cortex-M4 cores:

- The STM32MP15 is a heterogenous architecture that embeds up to two Cortex-A7 and one Cortex-M4
- The STM32MP13 embeds a single Cortex-A7

In this presentation, the icons in the top left corner will be used to show if the current slide is applicable to one or both part numbers.

When there are differences to highlight, the STM32MP15 will be shown first, since it was the first part number launched within the STM32MP1 family, and the STM32MP13 changes will be highlighted just after.



Platform low-power mode selection is the combination of the CPU and peripheral states.
In this slide, the Cortex-A7 CPU is shown in the middle, with frequency ranges below, whereas the STM32MP13 peripherals are shown on the left and STM32MP15 ones on the right.
This view allows a comparison of the low power modes for both platforms.

On the STM32MP1 series, the Cortex-A7 frequency can be up to 650 MHz with the nominal voltage, and above 650 MHz with the overdrive voltage.
Refer to the respective datasheets on the products to find the values of these voltages and the maximum frequencies for each part number.
Note that the software can implement “dynamic voltage and

frequency scaling", or DVFS, to dynamically adjust the CPU frequency according to the CPU load. This means that it will go to the overdrive mode when a huge amount of processing is going on and go back to the nominal mode otherwise. In this approach, each frequency and its associated voltage is referred to as an operating point or OPP.

The DVFS allows power savings in 'Run' mode and then, when there are no more tasks to be executed, the application can decide to put the platform in one of the available low power modes:

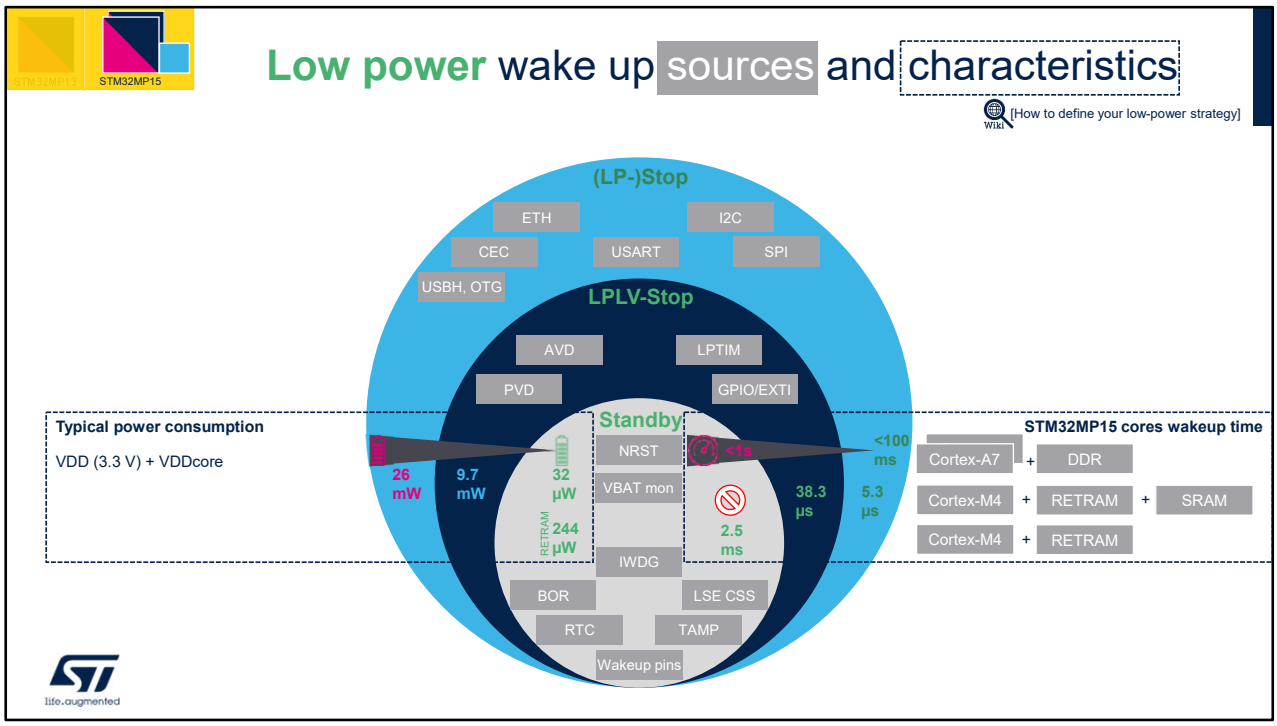
- The first level of sleeping corresponds to the various 'Stop' modes, where the CPU is simply clock gated but all supplies remain enabled, with the external regulator in 'low power' or even 'low voltage' mode to save a maximum of energy,
- Note that the STM32MP13 has another 'Stop' mode level with the 'LPLV-Stop2', which allows the CPU to be power-gated while the rest of the peripherals remain supplied,
- Further down in the low power mode ladder, we have the 'Standby' mode,
- And then, there is the 'VBAT' mode, and lastly, the 'Off' mode where all supplies are switched off.

Low power mode selection is driven by different factors, the first one being the peripherals we want to keep active and the ability to wake up the CPU when needed.

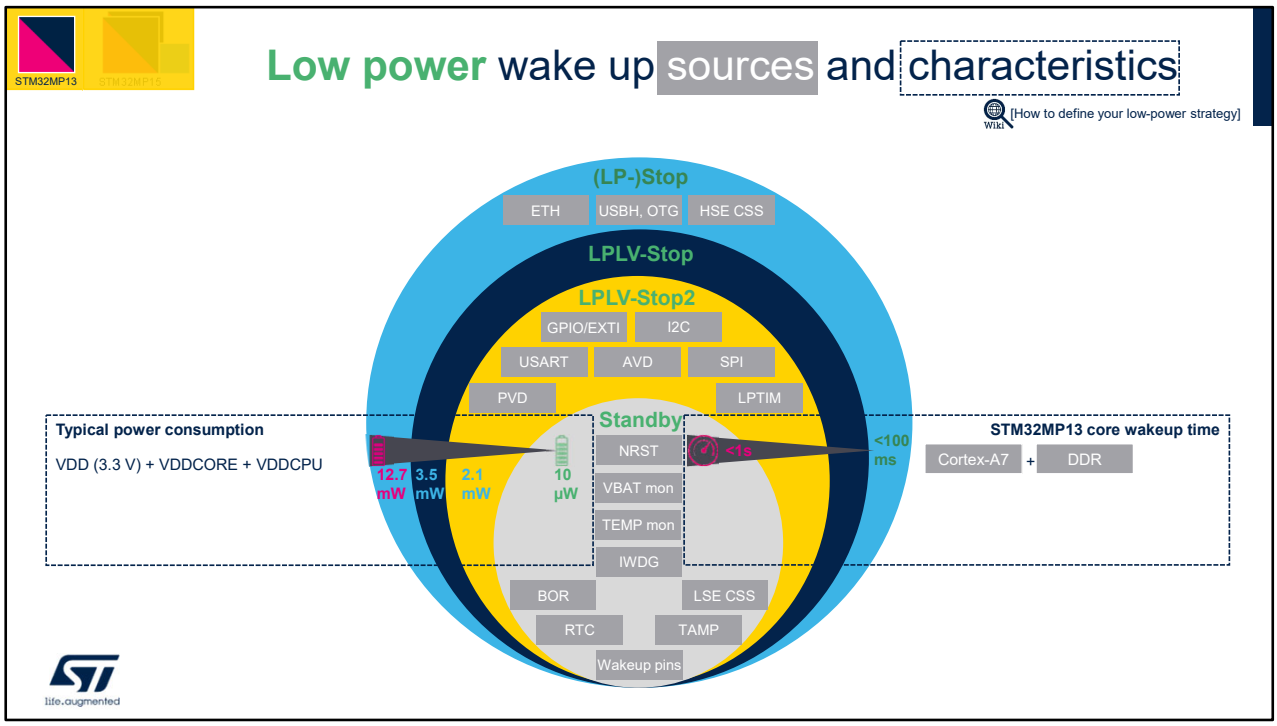
Each peripheral is supplied by one or more supplies among 'VDD', 'VDDCORE' and 'VBAT'.
STM32MP13 also has a 'VDDCPU' voltage domain dedicated to the CPU, which allows the 'LPLV-Stop2' mode.
Refer to the reference manual, datasheet and STM32 MPU wiki for further information on these supplies.

So, according to the peripherals we want to keep active, we can deduce the required power supply we need to keep on, and this will define the ultimate reachable low power mode.

Once this mode is found, the application may nevertheless decide to stay one step above in the ladder, since the final choice will be a compromise between the desired power consumption and the acceptable wake-up latency: “the nearer we are to the bed, the harder it is to run again!” This dilemma is illustrated in the slides that follow…

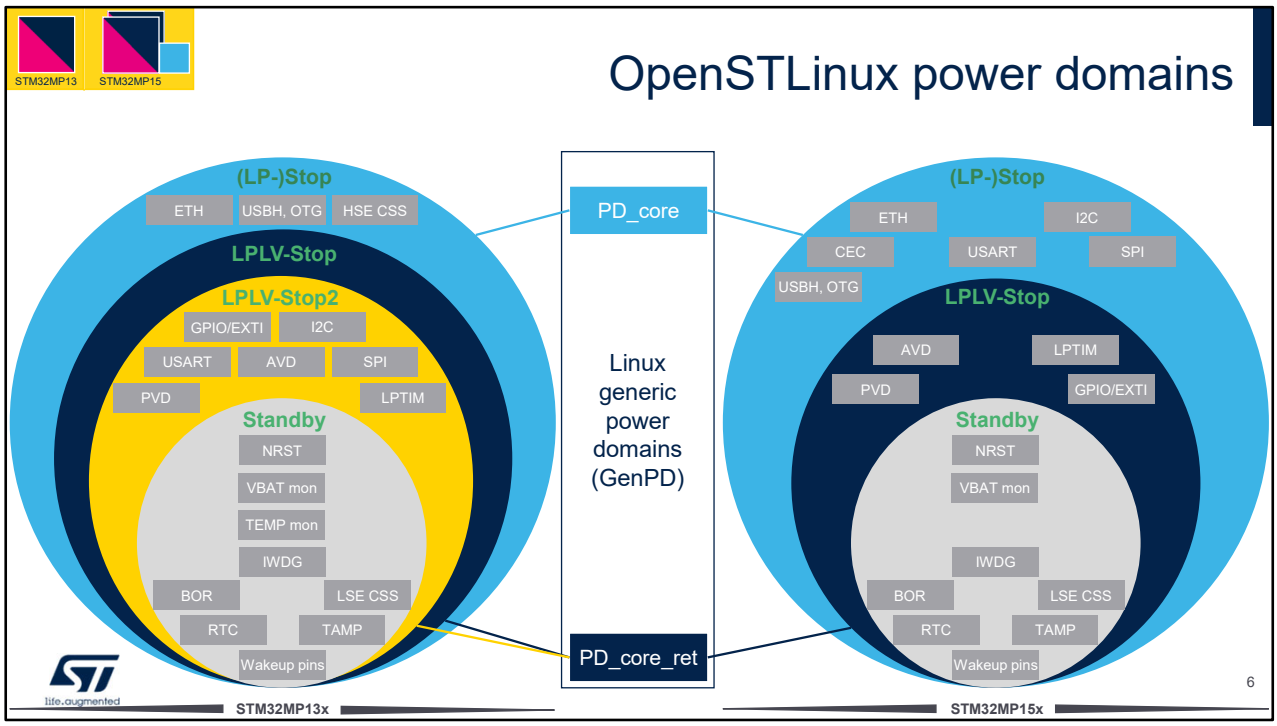


This slide shows the peripherals that can be a wake-up source for each low power mode supported by the STM32MP15.
Then it gives information on the die-level power consumption on the left, and the typical wake-up time duration, on the right.
Notice that the wake-up time depends on the software configuration so please refer to STM32 MPU Wiki for more precise figures and conditions.



This slide shows the peripherals that can be wake-up sources for each low power mode supported by the STM32MP13.
Then, it gives information on the die-level power consumption, on the left, and typical wake-up time duration, on the right.
Note that the wake-up time depends on the software configuration so please refer to STM32 MPU Wiki for more precise figures and conditions.

Compared to the STM32MP15, the STM32MP13 offers more wake-up source capabilities in ‘LPLV-Stop’ mode, and it introduces the ‘LPLV-Stop2’ mode to get even lower in power consumption with a negligeable wake-up overhead, mainly corresponding to VDDCPU rising time.



OpenSTLinux relies on the generic power domains framework and declares two power domains:

- ‘PD_core’ owns the peripherals corresponding to ‘Stop’ and ‘LP-Stop’ modes, like the ETH, in the light blue area.
- ‘PD_core_ret’ owns the peripherals corresponding to ‘LPLV-Stop’, in dark blue, and ‘LPLV-Stop2’, in yellow, like the LPTIM. The latter is only available on STM32MP13.

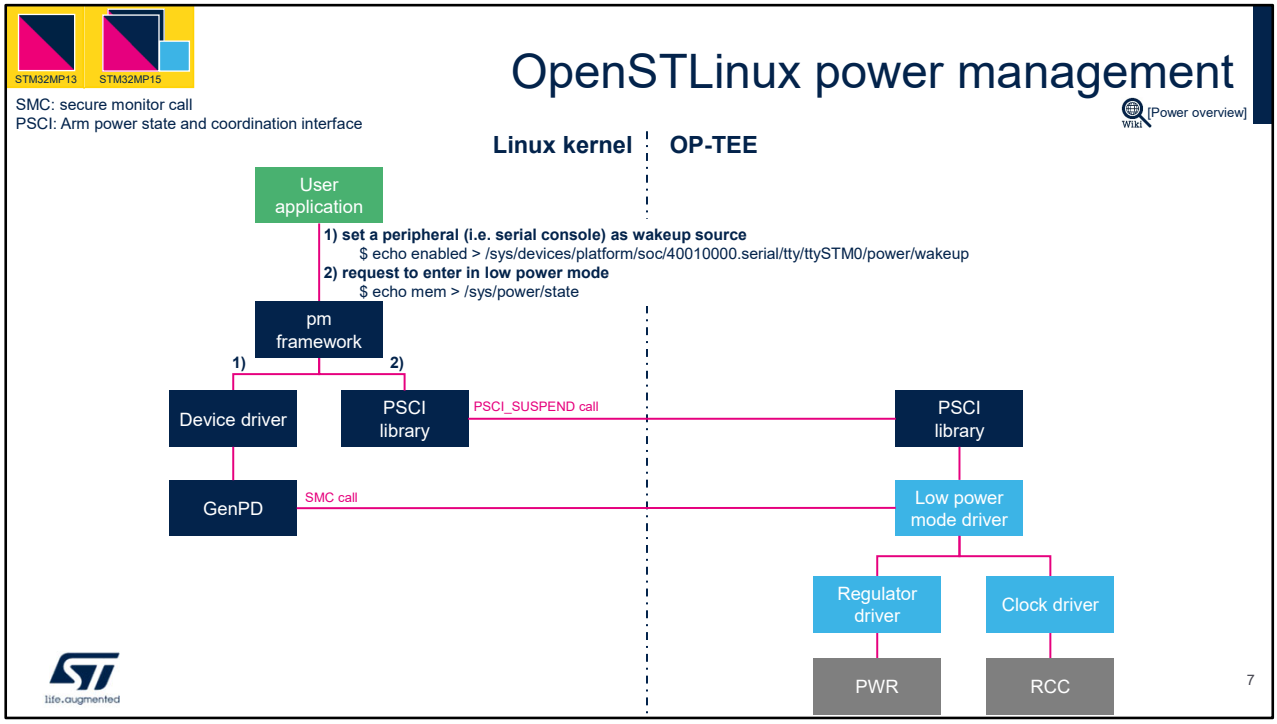
Based on this mapping, when the application sets a peripheral as wake-up source, the Linux kernel requests the Secure OS, OP-TEE that the corresponding power domain not be switched off.

Later, when the application requests entry into low power mode, the Linux kernel asks OP-TEE to suspend the platform and OP-TEE knows which low power mode it can

reach according to the power domain that must be kept enabled.

For instance, if the ETH is set as wake-up source, then the PD_core must not be switched off and the platform is only able to enter ‘Stop’ or ‘Lp-Stop’ mode.
If LPTIM is set as wake-up source, then the PD_core can be switched off and the platform can enter LPLV-Stop mode or even ‘LPLV-Stop2’ mode on STM32MP13.
Note that the selection between ‘LPLV-Stop’ and ‘LPLV-Stop2’ mode is an OP-TEE compilation choice.

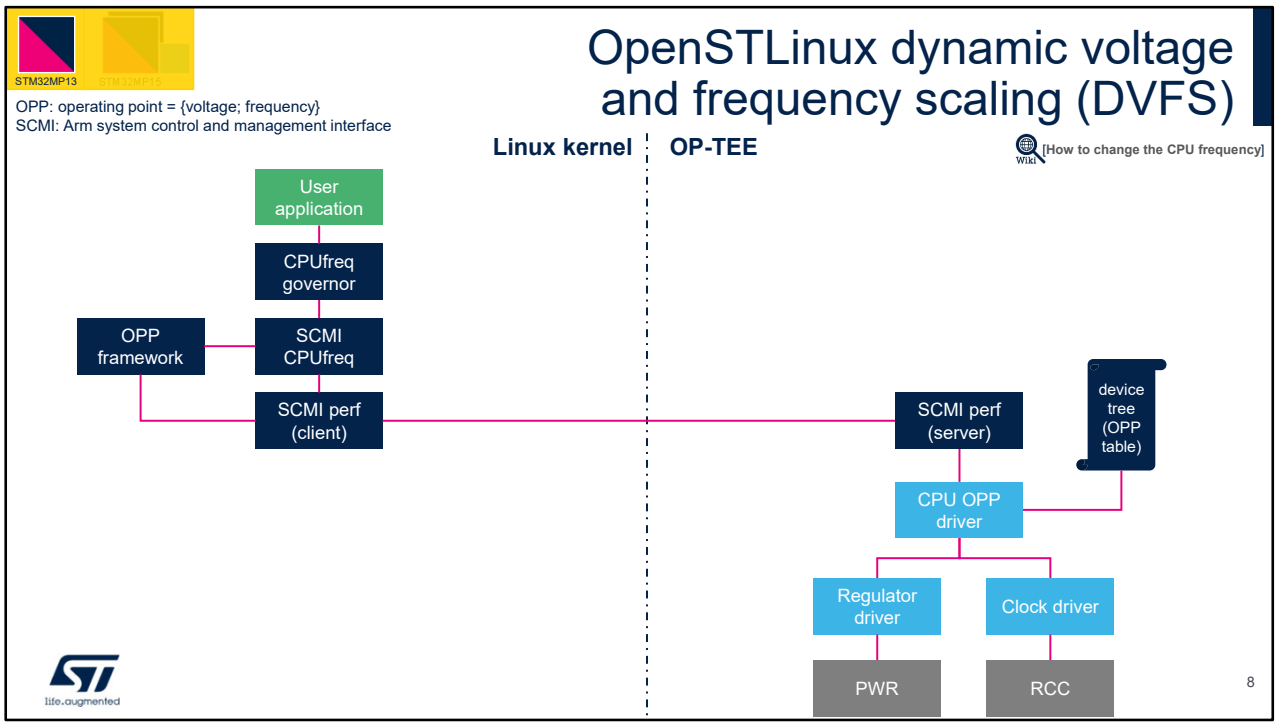
The following slides give more details on the underlying architecture and application dynamics.



Putting the platform in low power mode takes place in two steps:

- first, the Linux application selects the desired wakeup source. The given command line example allows the serial console to be set as wake-up source.
- then, the application requests to enter low power mode via the given and unique ‘echo mem’ command.

The selected wakeup source allows it to determine which power domain must be kept on and therefore, OP-TEE can finally select the ultimate low power mode.



The operating points, or OPPs, are declared in the OP-TEE device tree and discovered at boot time by Linux.

Then, the selected Linux CPUfreq governor will dynamically request OP-TEE to jump to the upper or lower OPP at runtime, depending on the CPU load, via the SCMI performance framework.

Thank you

life.augmented