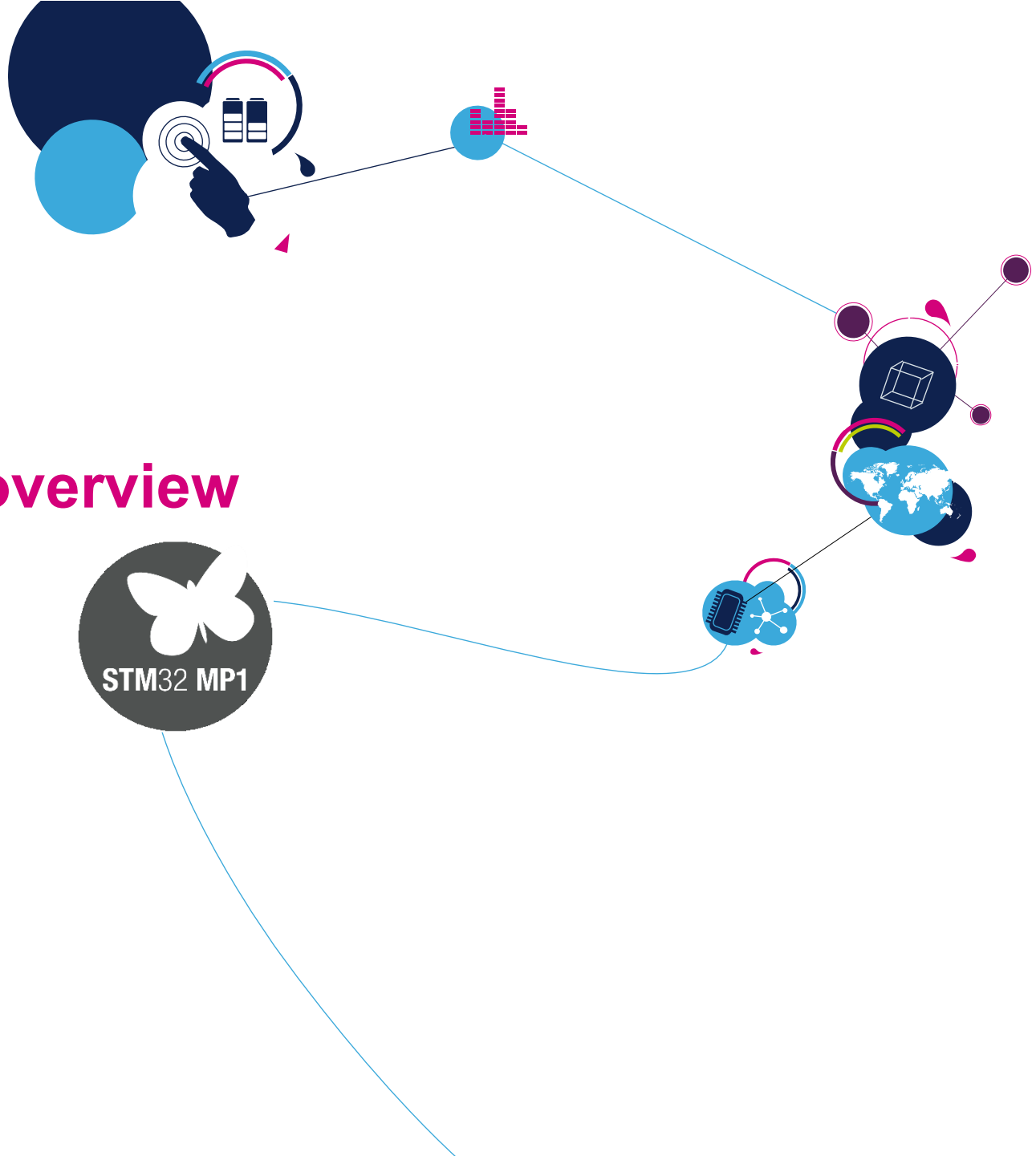
STM32MP1

Coprocessing Management overview



life.augmented

Agenda

Presentation

30min

- M4 Firmware Loading
- Multi-core Resources management
- Shared resources management

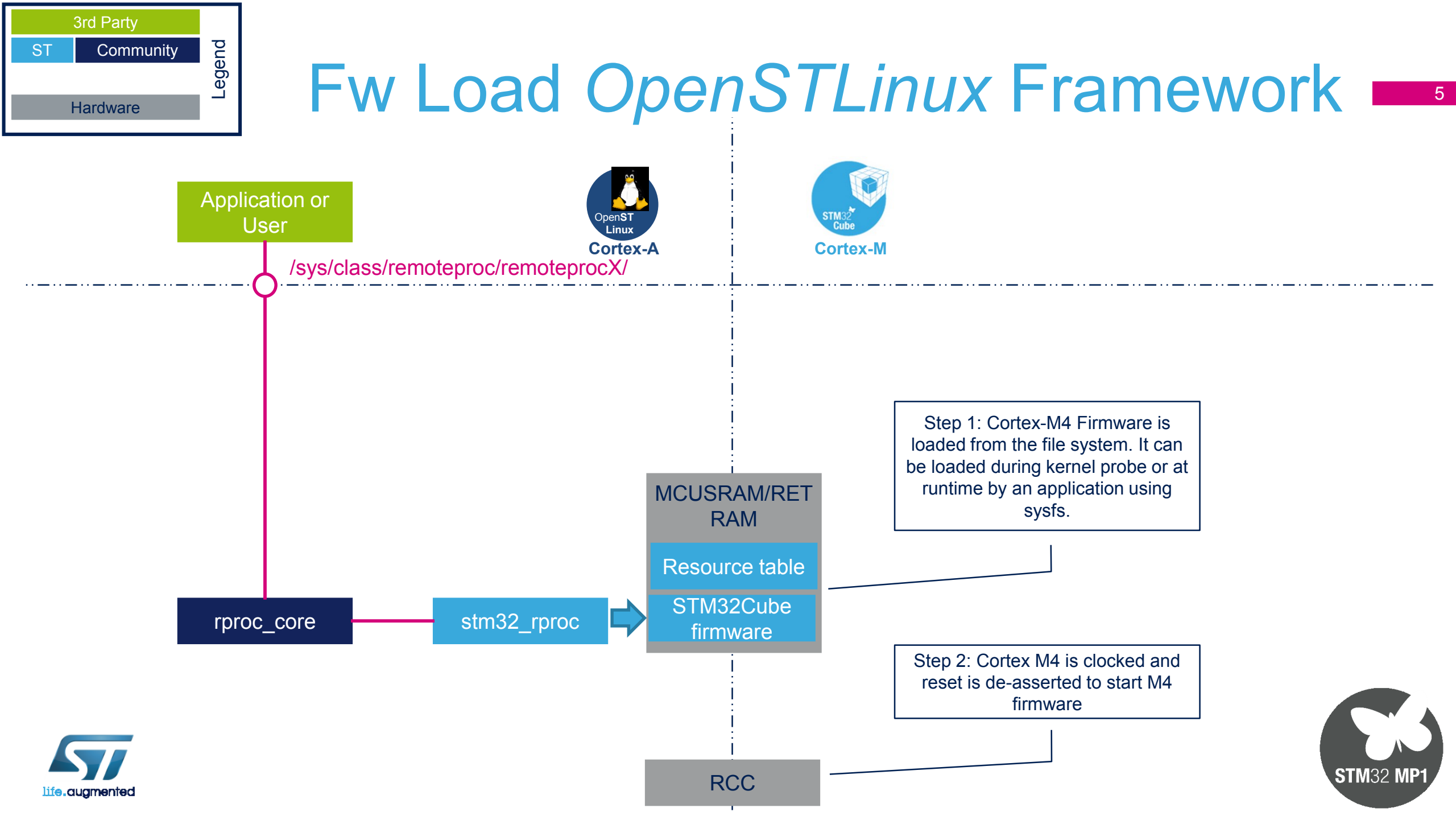
Cortex-M4 advantage

- Save energy
- Real time constraints
- Offload the A7
- Provide more peripheral accessible
- Reuse of already developed firmware on MCU

M4 Firmware loading

- The Cortex-M4 firmware is loaded in *MCURAM* by Cortex-A7
- Load of the firmware is done either :
  - **Automatically by Uboot** (early boot)
  - **Automatically by OpenSTLinux kernel**
  - **Manually** at OpenSTLinux runtime
- In OpenSTLinux uses the *RemoteProc* framework
- Firmware is stored in an ELF format file in OpenSTLinux filesystem (/lib/firmware)

Fw Load *OpenSTLinux* Framework



life.augmented

STM32 MP1

Multi-core Resources management

- M4 is seen as a coprocessor of A7
  - Peripherals are assigned to A7 or to M4
  - *OpenSTLinux* controls Clock Tree and Power regulator
  - M4 firmware controls its peripherals independently from A7 (except clock frequency & regulators)
  - For M4 assigned peripheral, OpenSTlinux *rproc-srm* (system resource manager) ensures reservation of peripheral Clocks and GPIOs.

Peripheral Assignment

Peripherals	A7S	A7NS	Cortex-M4
I2C5		☐	☐
UART4		☑	☐
UART5		☐	☑
UART7		☐	☑
UART8		☐	☐
USART2		☐	☐
USART3		☐	☐
USART6		☐	☐
SPI1		☑	☐
SPI2		☑	☐
SPI3		☐	☐
SPI4		☐	☐
SPI5		☐	☐
QUADSPI		☐	☐
SDMMC1		☑	
SDMMC2		☑	
SDMMC3		☐	☐
USBH_HS1		☑	
USBH_HS2		☐	
USB_OTG_HS		☑	☐
I2S1		☐	☑

- All the peripherals is assigned to one hw execution context
- Peripheral Assignment configuration is stored in the TF-A device tree file
- STM32CubeMX supports assignment. TF-A device tree file is generated by the tool.

A7-M4 Shared resources management

- The A7 OpenSTLinux has the control of the *clock tree frequencies* and the *regulators*.
  M4 can ask OpenSTLinux resource manager for a dynamic reconfiguration

- The M4 firmware and OpenSTLinux on A7 shares some peripherals.
  - The GPIO clocks and GPIO pin muxing
  - The EXTI configuration
  - RCC for IP clk gating, IP reset

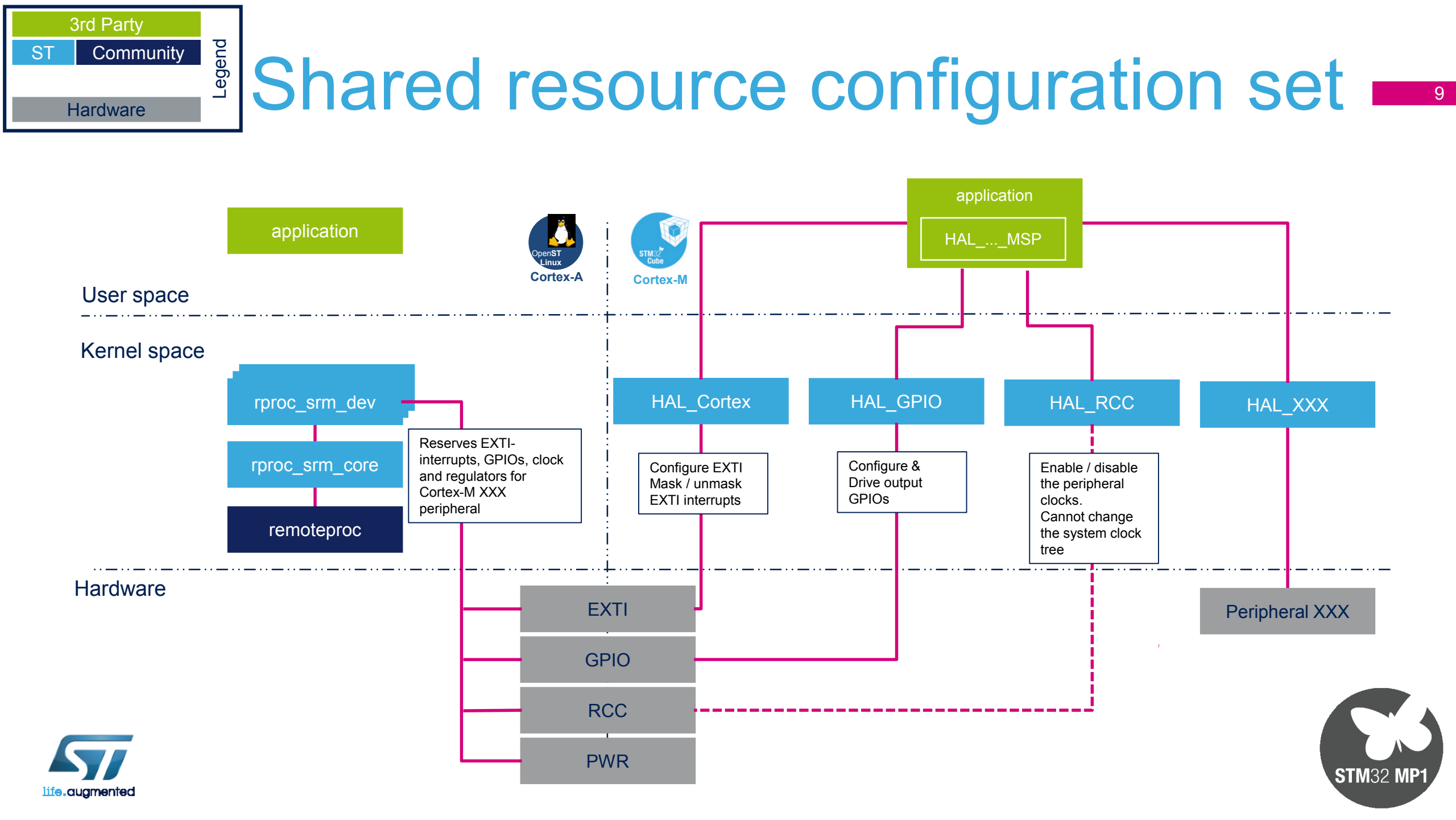
```
PERIPH_LOCK(GPIOA);
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
PERIPH_UNLOCK(GPIOA);
```

- GPIO&EXTI configuration is via HSEM h/w semaphore => avoids concurrent access (rproc-srm on A7, HAL driver on M4)

- RCC IP (clock & reset) is via MCU and MPU dedicated registers - HSEM h/w semaphore not required

- Configuration of SYSCFG,DMAMUX,IPCC peripherals requires HSEM h/w semaphore

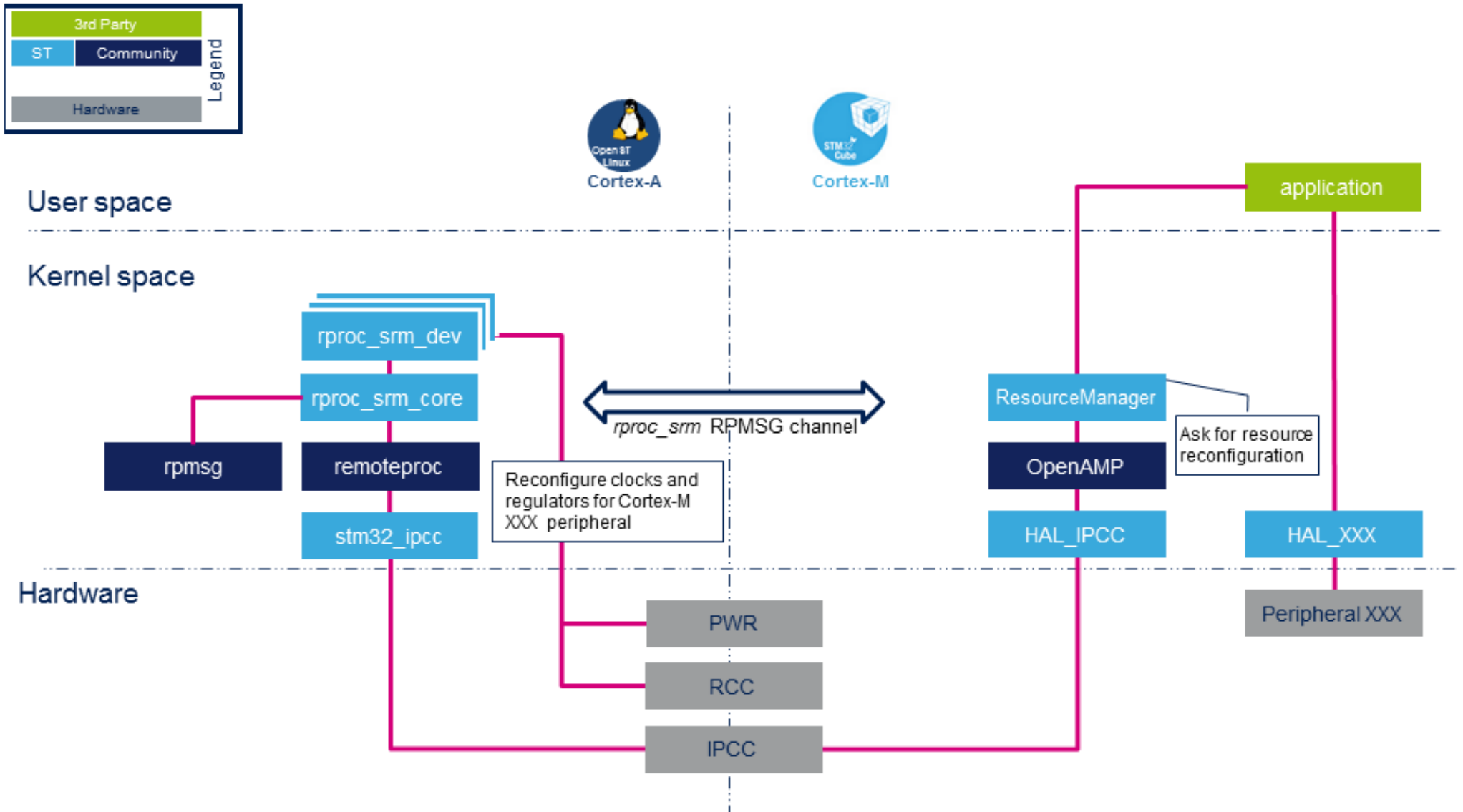
Shared resource configuration set



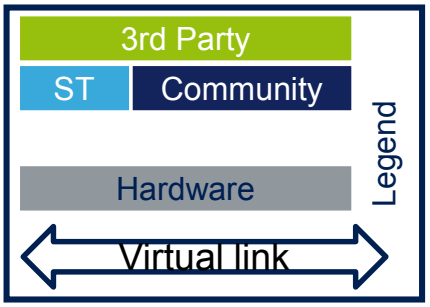
life.augmented

STM32 MP1

Dynamic system resource update



Coprocessor software framework



Inter-processor communication (I2C variant exists too)

application

/dev/ttyRPMSG0

stm32_rpmsg_tty

rpmsg

rproc_core

stm32_ipcc

OpenSTLinux Distribution

Cortex A

STM32 Cube

Cortex-M

TTY RPMSG channel

MCUSRAM

RX & TX Vrings

Buffers

IPCC

application

virtual_hal_uart

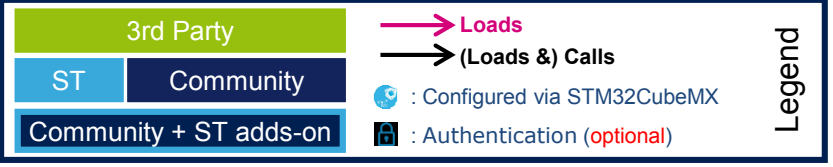
OpenAMP

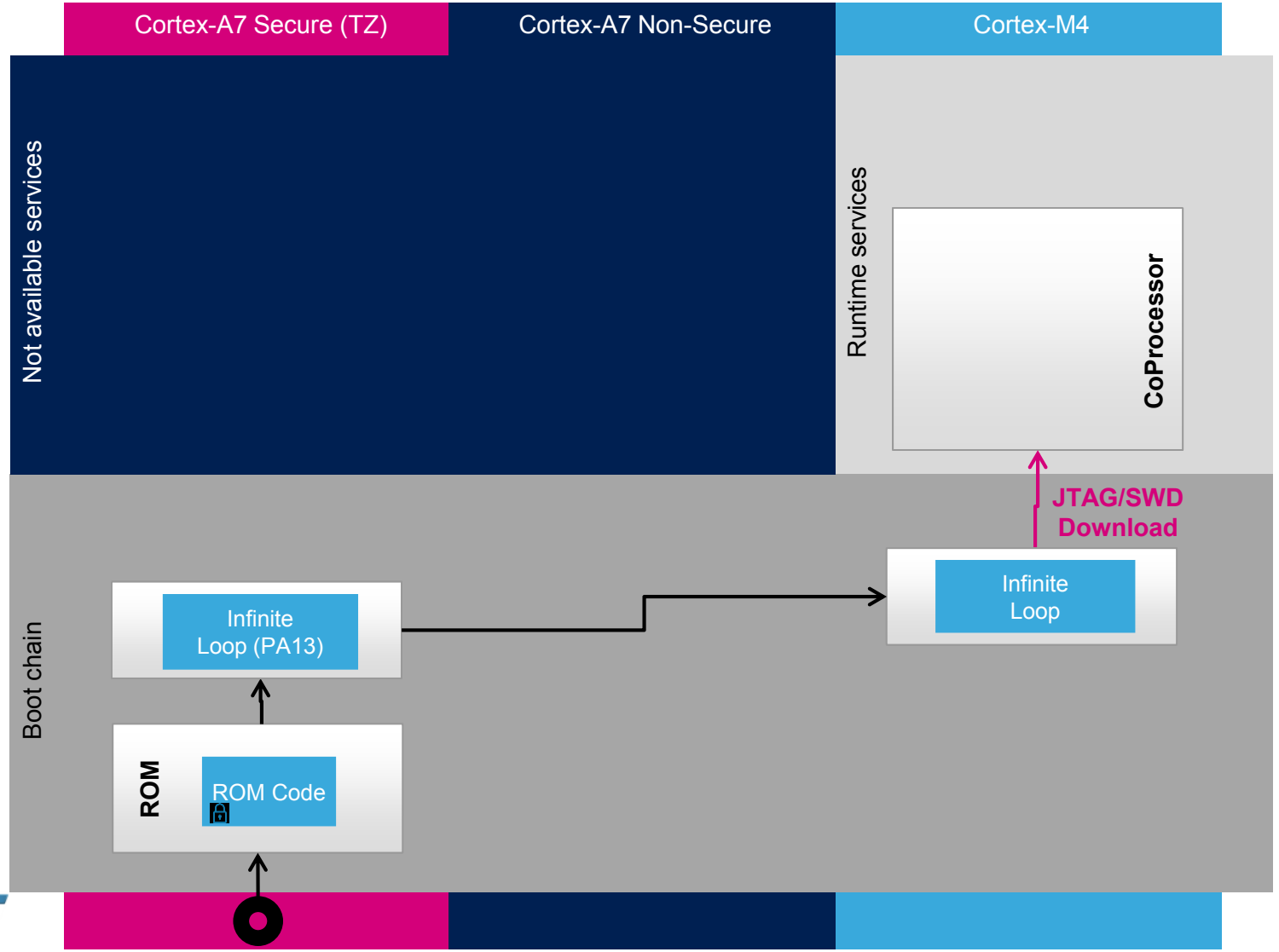
HAL_IPCC

life.augmented

STM32 MP1

Engineering Boot chain





- Used in M4-only preliminary debug – no Linux boot image needed
- M4 Firmware Load is done via JTAG/SWD
- Need to address clocks and pio alternate function set-up

Boot Pins: “0b100”

life.augmented

STM32 MP1

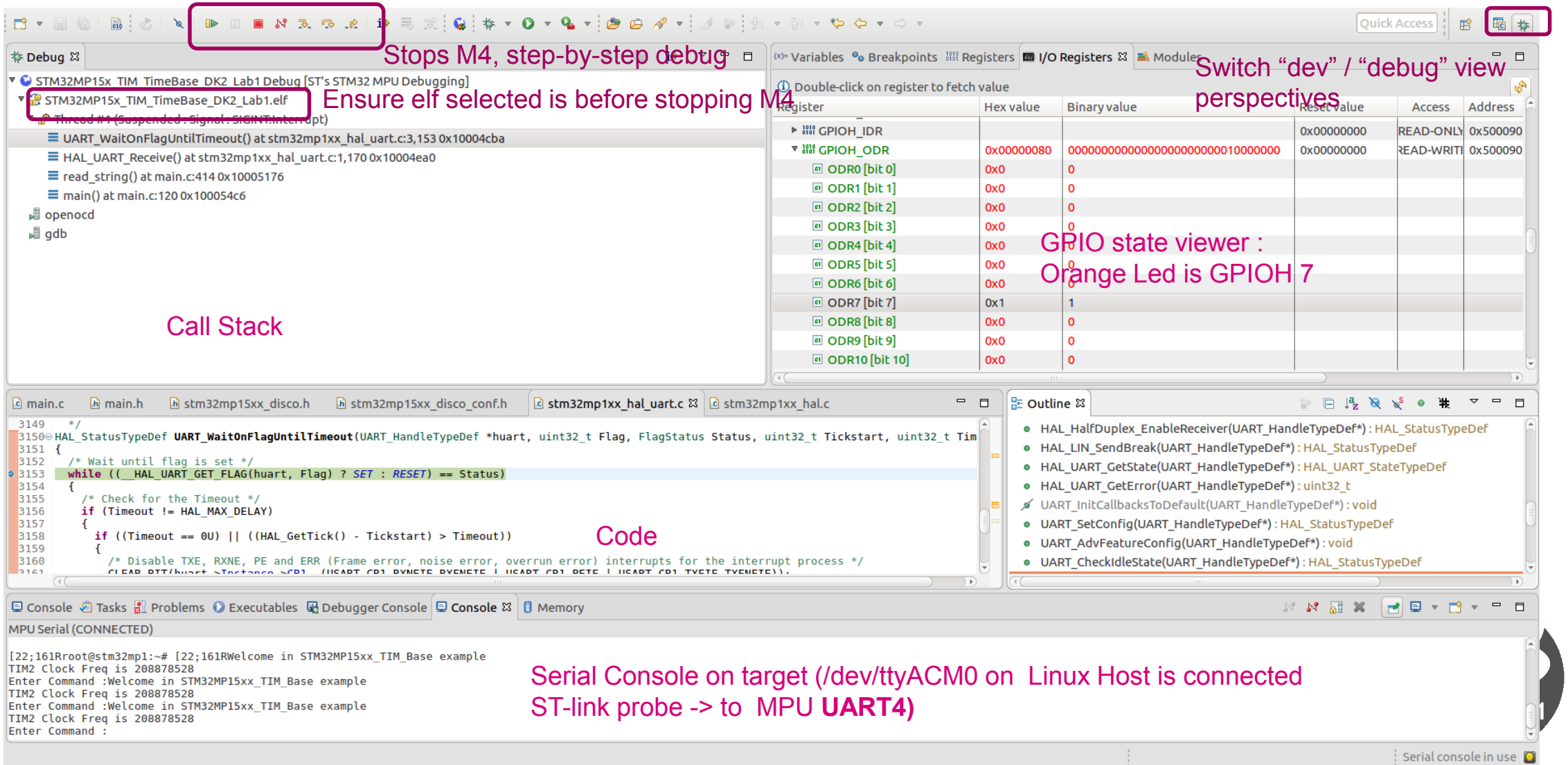
M4 Debug in Engineering mode

- Used in M4-only preliminary debug
  - No Linux boot image needed
- M4 Firmware Load is done via JTAG/SWD
- Need to address clocks and pins alternate function set-up

Boot Pins: “0b100”

M4 Debug with Linux (CoProc Eclipse plug-in)

Debug perspective



life.augmented