STM32MP1 platform boot

life.augmented

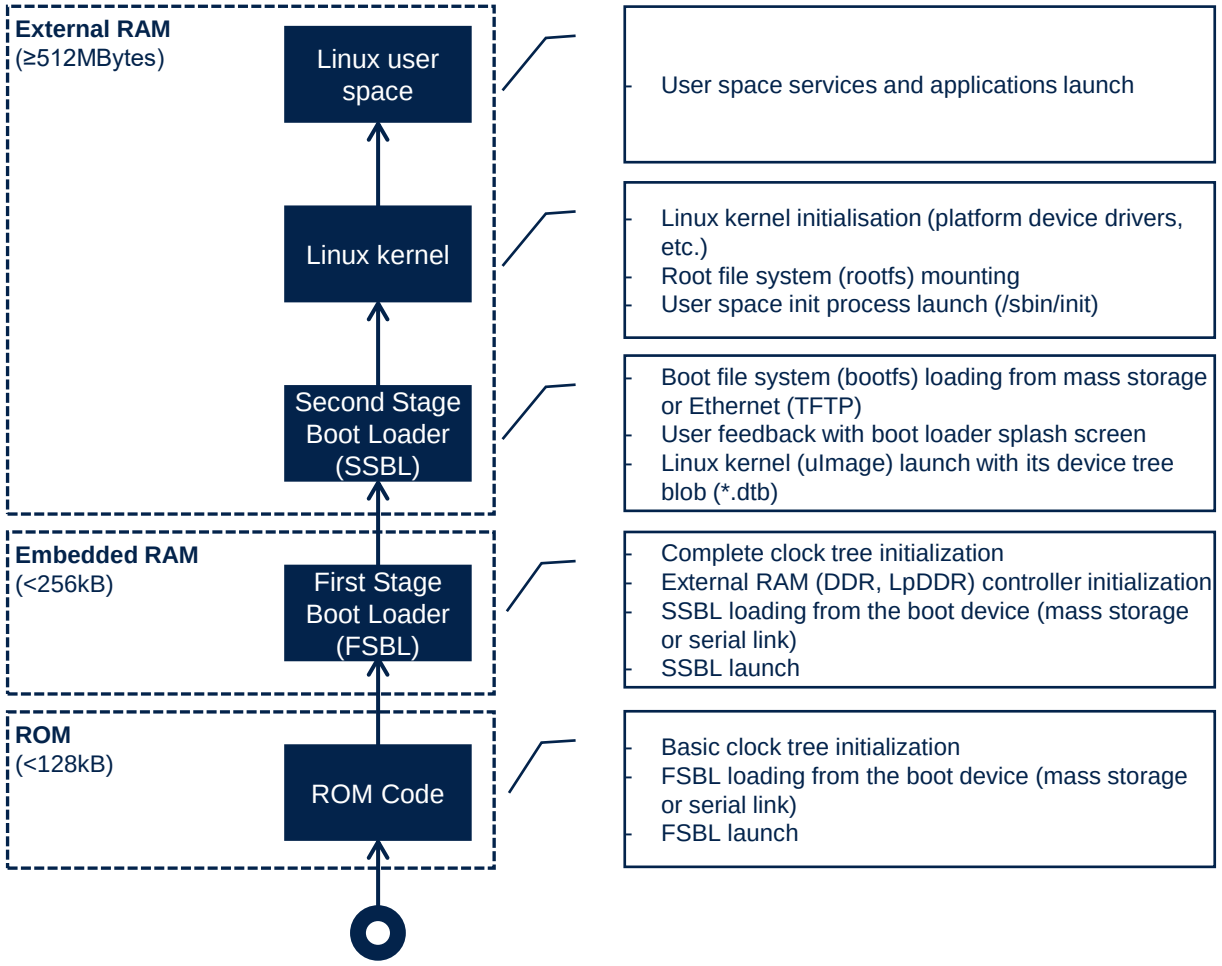
Content

- Boot chain overview
  - Boot stages
  - Boot mode selection
  - Flash programming
- Boot chain configuration
- OpenSTLinux flash memory mapping
- Appendix
  - Code authentication

Boot chain overview

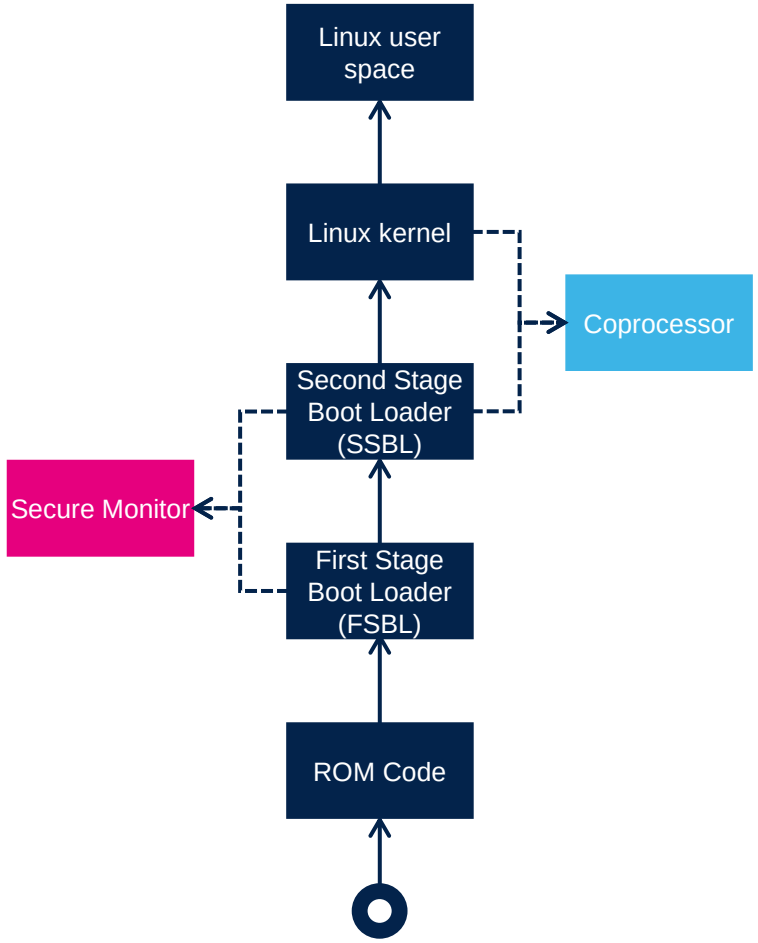
Standard linux boot chain

Rocket image source:
https://fr.wikipedia.org/wiki/Ariane_6

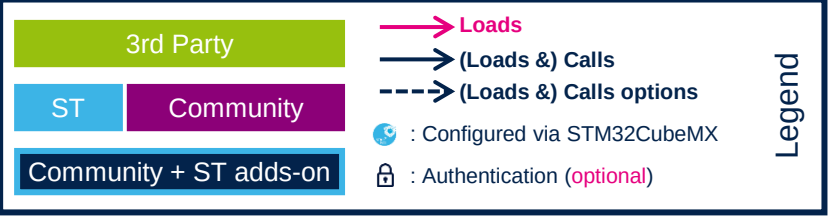


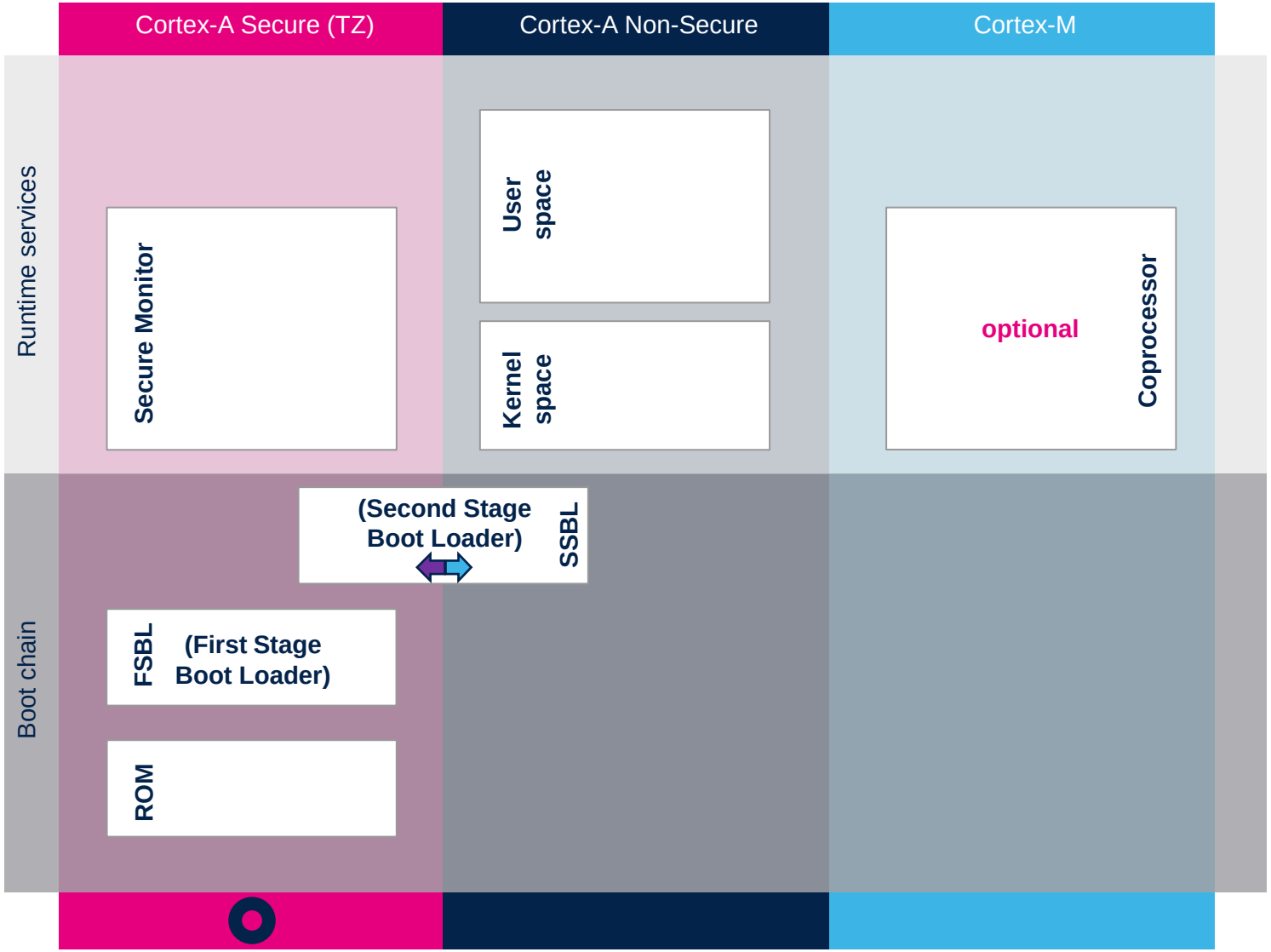
life.augmented

STM32MP1 boot chain



STM32MP1 boot chains

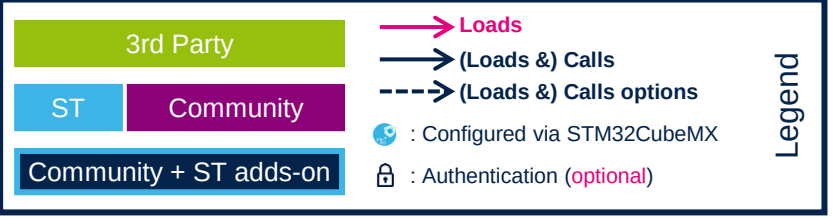


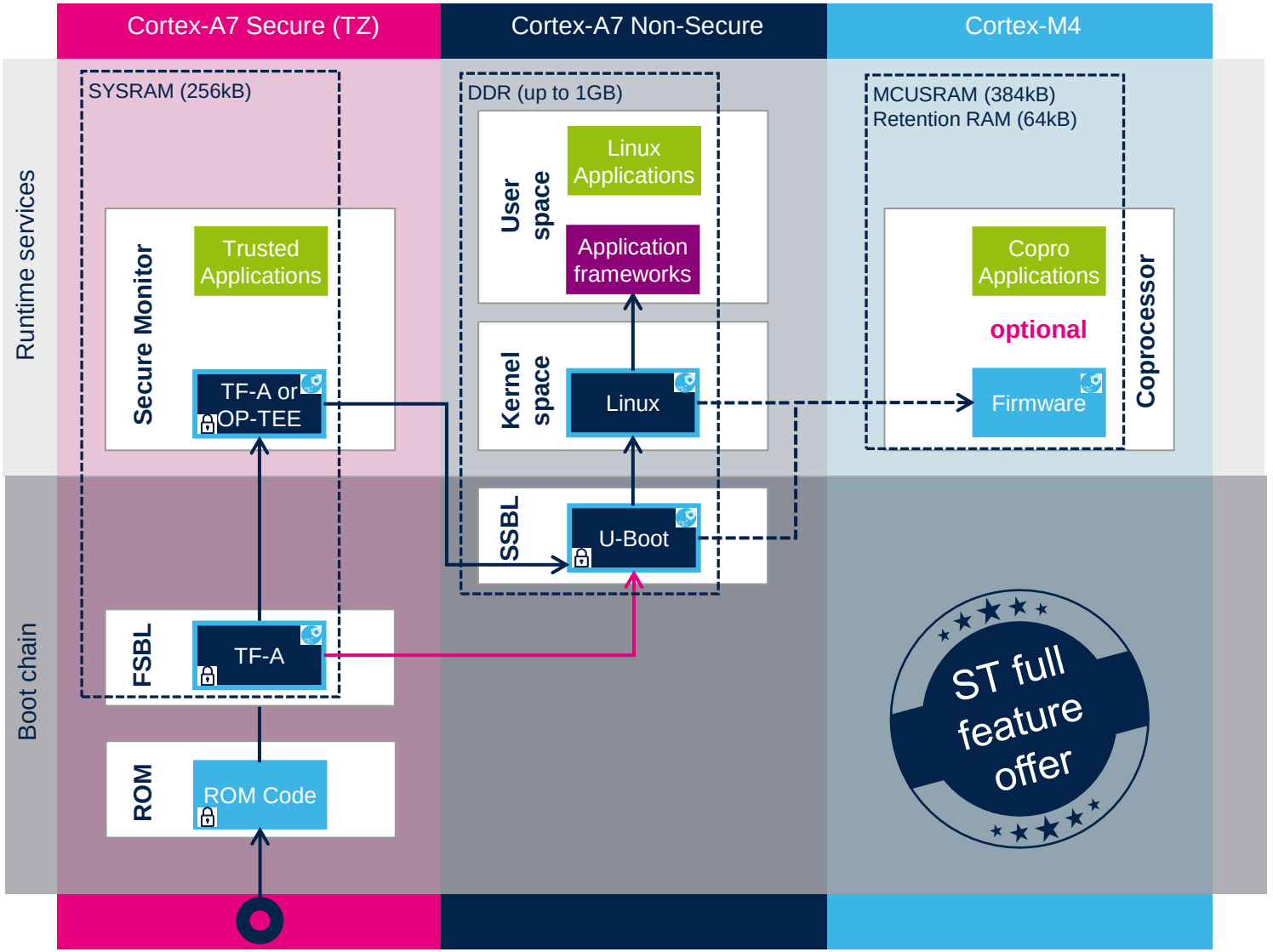


life.augmented

: the Second Stage Boot Loader (SSBL) can run in ARM Secure/Trustzone mode (for Basic boot) or Non-Secure mode (for Trusted boot)

Trusted boot chain



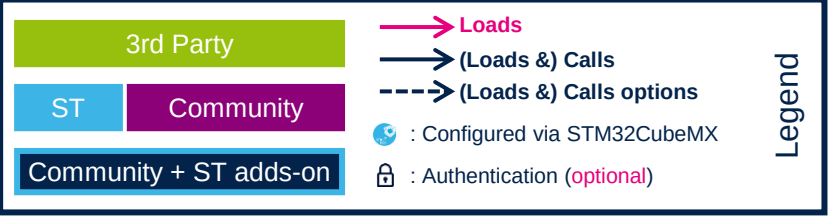


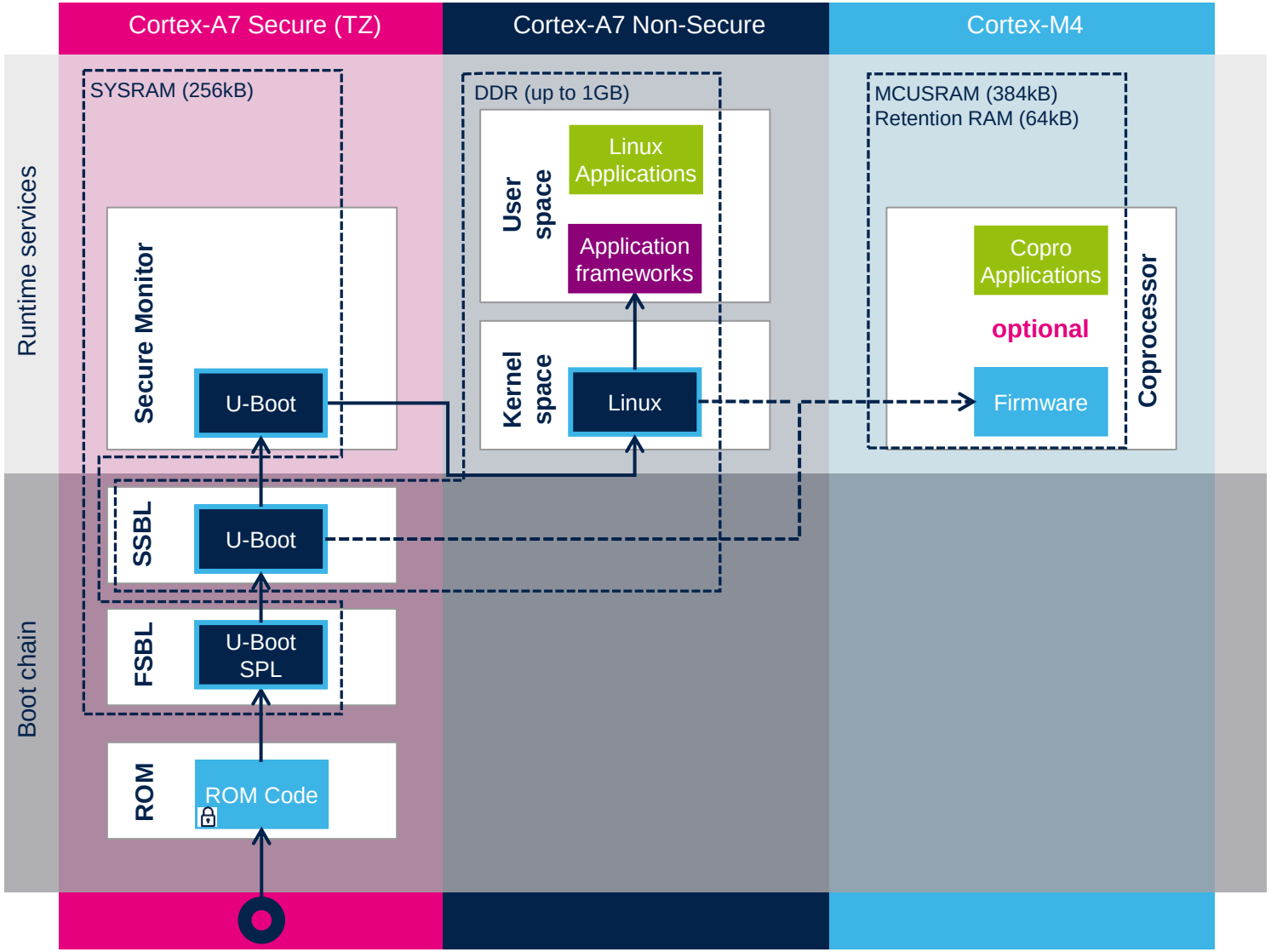
TF-A is:
-BSD licence
-Trusted writing
-ARMv8 future proof

Secure Monitor: Trusted Firmware for Cortex-A (TF-A) Secure Monitor is used if there is no Secure OS (OP-TEE is optional)

life.augmented

Basic boot chain





Boot mode selection

BOOT pins	TAMP_REG[20] (Force Serial)	OTP WORD 3 Primary boot source	OTP WORD 3 Secondary boot source	Boot source #1	Boot source #2 if #1 fails	Boot source if #2 fails
b000	x (don't care)	x (don't care)	x (don't care)	Serial	-	-
b001	!= 0xFF	0 (virgin)	0 (virgin)	QSPI NOR	Serial	-
b010	!= 0xFF	0 (virgin)	0 (virgin)	eMMC	Serial	-
b011	!= 0xFF	0 (virgin)	0 (virgin)	FMC NAND	Serial	-
b100	x (don't care)	x (don't care)	x (don't care)	NoBoot	-	-
b101	!= 0xFF	0 (virgin)	0 (virgin)	SD-Card	Serial	-
b110	!= 0xFF	0 (virgin)	0 (virgin)	Serial	-	-
b111	!= 0xFF	0 (virgin)	0 (virgin)	QSPI NAND	Serial	-
!= b100	!= 0xFF	Primary[1]	0 (virgin)	Primary[1]	Serial	-
!= b100	!= 0xFF	0 (virgin)	Secondary[1]	Secondary[1]	Serial	-
!= b100	!= 0xFF	Primary[1]	Secondary[1]	Primary[1]	Secondary[1]	Serial
!= b100	0xFF	x (don't care)	x (don't care)	Serial	-	-

0	No secondary boot source is defined
1	FMC NAND

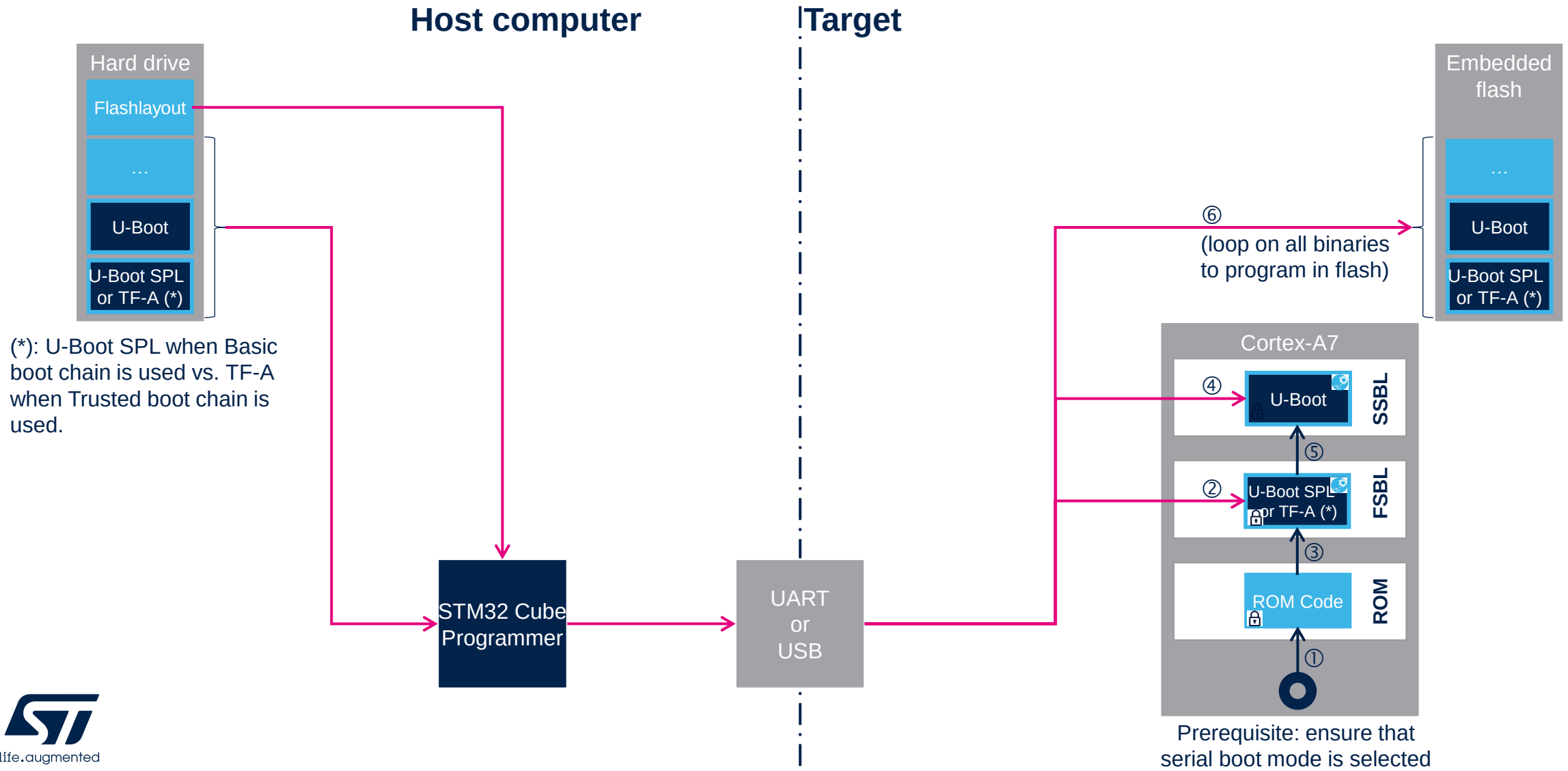
0	No primary boot source is defined
1	FMC NAND
2	QSPI NOR
3	eMMC
4	SD
5	QSPI NAND

[1]Primary and Secondary are fields of OTP WORD3.

life.augmented

Reference: AN5031 - Getting started with STM32MP15 Series hardware development.
Wiki article: [STM32MP15_ROM_code_overview]

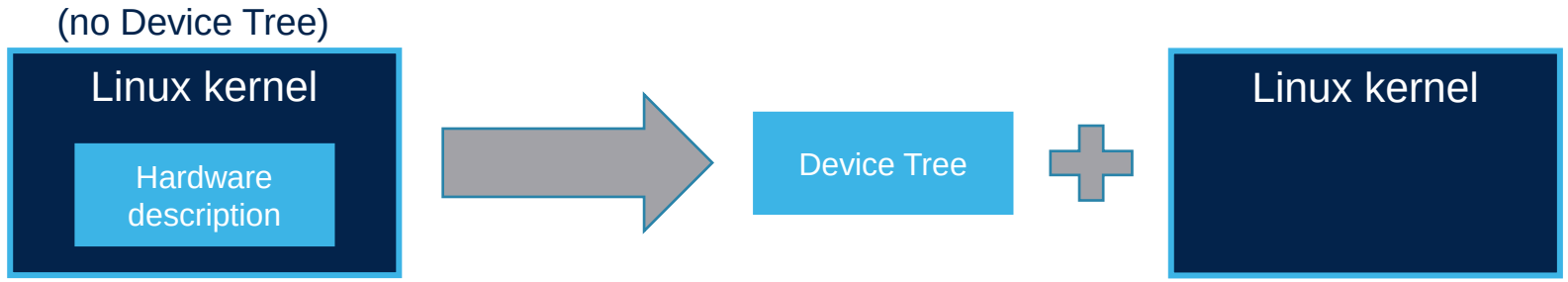
Stm32cubeprogrammer for flash programming



Boot chain configuration

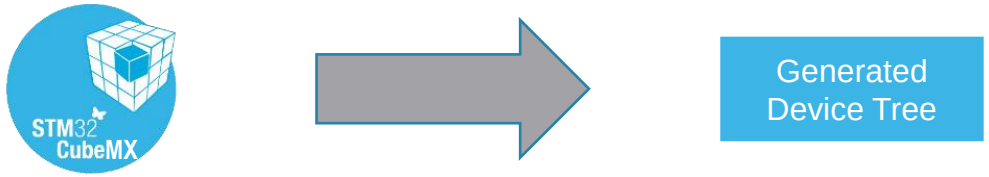
Variability management via device tree

- Former **Linux** kernel used to embed the hardware description of the supported board in the same binary. Current kernels put this information in a separate binary, the **device tree blob** (dtb). As a consequence, a unique kernel binary can support different chips and boards. **U-Boot** also adopted the same solution.



- Device Tree documentation is available on http://elinux.org/Device_Tree, starting from Device Tree for Dummies from Thomas Petazzoni.

- Linux developers manually edit device tree source files (dts): STMicroelectronics enables this **generation from STM32CubeMX** to ease new comers hands-on!



Device tree example for STM32MP1

stm32-usart.c

```
static const struct of_device_id stm32_match[] = {
        { .compatible = "st,stm32h7-uart", .data = &stm32h7_info},
```

stm32mp157c.dtsi

```
uart4: serial@40010000 {
    compatible = "st,stm32h7-uart";
    reg = <0x40010000 0x400>;
    interrupts-extended = <&intc GIC_SPI 52 IRQ_TYPE_NONE>, <&exti 30 1>;
    clocks = <&rcc_clk UART4_K>;
    status = "disabled";
};
```

stm32mp157-pinctrl.dtsi

```
uart4_pins_a: uart4@0 {
    pins1 {
    pinmux = <STM32_PINMUX('G', 11, AF6)>; /* UART4_TX */
    bias-disable;
    drive-push-pull;
    slew-rate = <0>;
};
…
```

stm32mp157c-ed1.dts

```
&uart4 {
        pinctrl-names = "default";
        pinctrl-0 = <&uart4_pins_a>;
        status = "okay";
};
```

life.augmented

U-boot binaries vs. Execution contexts

- U-Boot source code leads to the generation of two binary files:

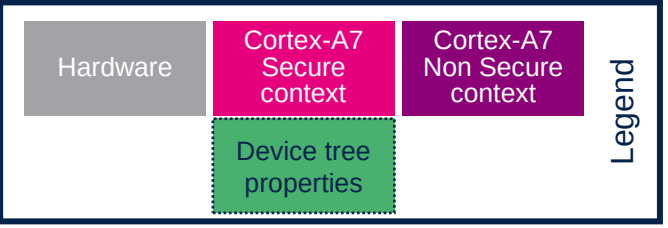
Binary	Execution context	Comment
u-boot-spl.bin	FSBL	Only applicable with the Basic boot chain, since the Trusted boot chain is using TF-A as FSBL.
u-boot.bin	SSBL pre-reloc	U-Boot is relocating itself from the beginning of the DDR, where it is loaded by the FSBL, to the end of the DDR: this defines the pre-reloc context.
	SSBL	

  - A **device tree blob** is appended at the end of each binary:
    - u-boot-spl.bin device tree is gotten via U-Boot **fdtgrep** tool
      - fdtgrep filters out all nodes that do not have "**u-boot,dm-spl**" or "**u-boot,dm-pre-reloc**" property to reduce it as much as possible for "FSBL" context (running in the narrow SYSRAM)
    - u-boot.bin device tree is not filtered but it will be used differently from each context:
      - "SSBL pre-reloc" only takes into account the nodes with "**u-boot,dm-pre-reloc**" property
      - "SSBL" context uses the complete device tree

U-boot configuration

- Build time configuration
  - Board definition
    - u-boot/include/configs/stm32mp*.h
      - memory mapping, boot command, features enabling (that are not in Kconfig)
  - U-Boot features
    - u-boot/configs/stm32mp*_defconfig
      - Target selection, features enabling (distro, bootdelay, spl, …)
      - Modified via menuconfig
- Runtime configuration
  - Device tree (cf. next slide)
    - Selected via defconfig or with make option DEVICE_TREE
    - Appended after U-Boot binary
    - U-Boot is a device tree consumer (for its needs) and provider (for Linux kernel)
  - DISTRO (see doc/README.distro)
    - Enabled via defconfig

Boot contexts



Peripherals initialization

Remaining		… (*)
Pinctrl	GPIO (boot devices, console, LEDs)	GPIO (*)
Serial console	UART4	
Power supply	PMIC, I2C4	
System time	STGEN	
Clock tree	PWR, RCC (PLLs setup)	PWR, RCC
DDR	DDR CTRL / PHY	
Boot devices	SDMMC, FMC, QSPI, UART, USART, OTG	idem + ETH, USBH
Firewalling	ETZPC, TZC	

Boot chain

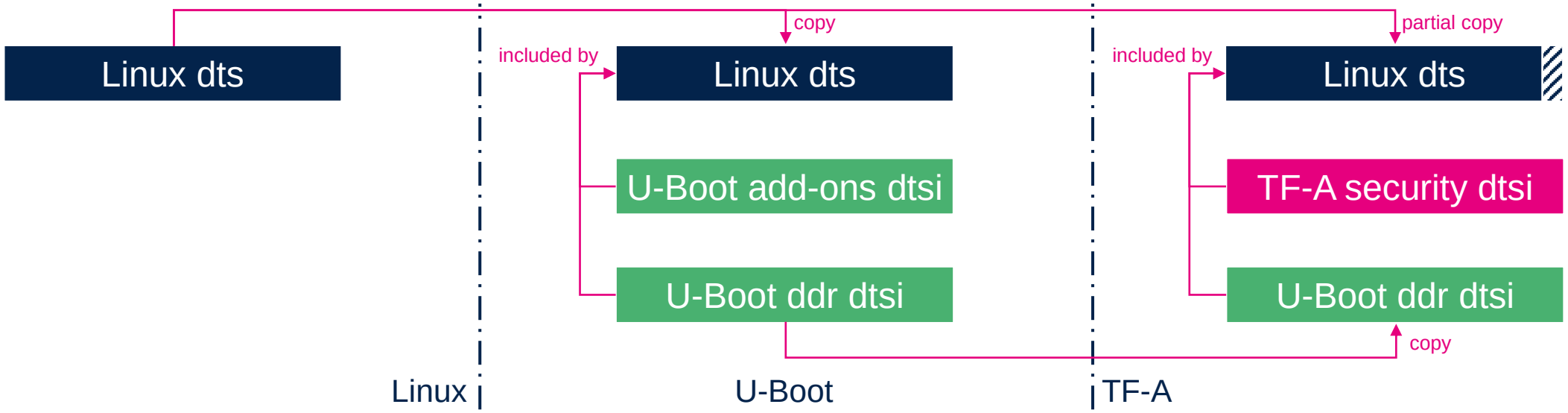
	ROM code	U-Boot pre-reloc	U-Boot	Linux
Trusted	TF-A			
	Remove useless nodes!			
	u-boot,dm-spl			
	u-boot,dm-pre-reloc			
Basic	U-Boot SPL			

Reset	ROM context	FSBL context	SSBL pre-reloc context	SSBL context	Runtime context

(*) : the same complete device tree can be given to U-Boot and Linux, but U-Boot may only probe a subset of all peripherals, depending on its build time configuration

life.augmented

Device tree for linux, u-boot & TF-A



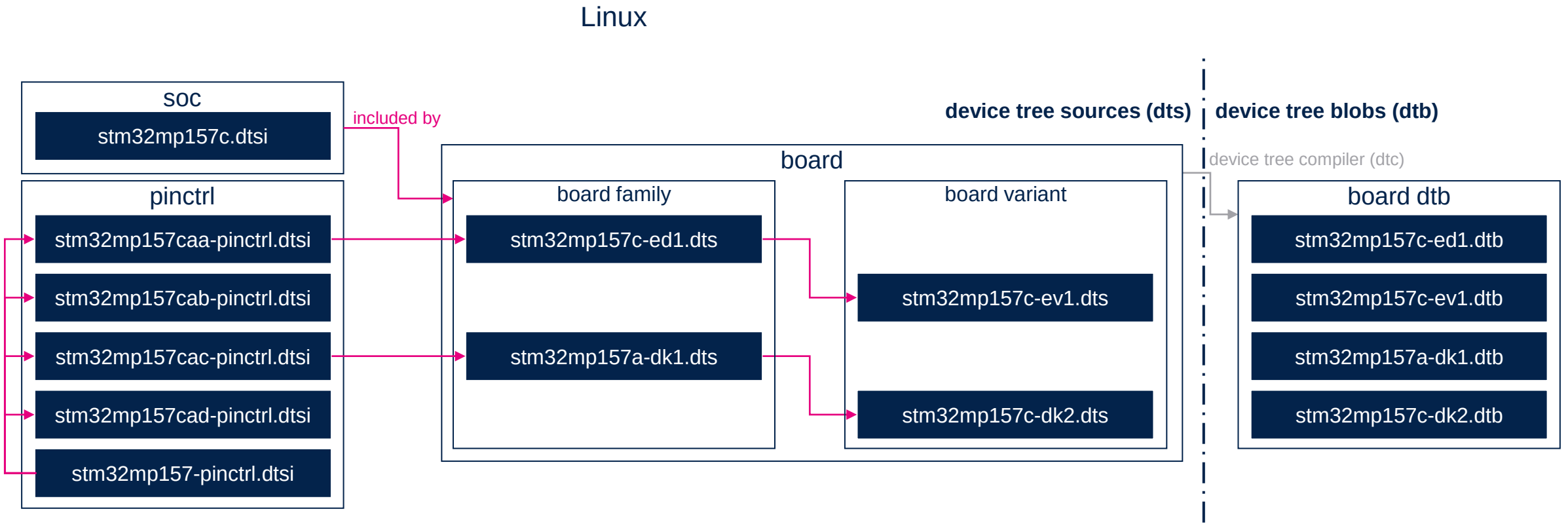
- In Linux, STM32MP15 is supported via a set of device tree source files (dts)
- In U-Boot, Linux dts files are copied and overloaded with U-Boot add-ons properties and DDR configuration
- In TF-A, Linux files are partly copied then completed with the DDR configuration (copied from U-Boot) and security configuration (firewalling)

life.augmented

- **ed: evaluation daughter board** (embedding the STM32MP1)
- **ev: evaluation board** (ed plugged on the mother board)
- **dk: discovery kit**

Legend

Upstreamed device tree



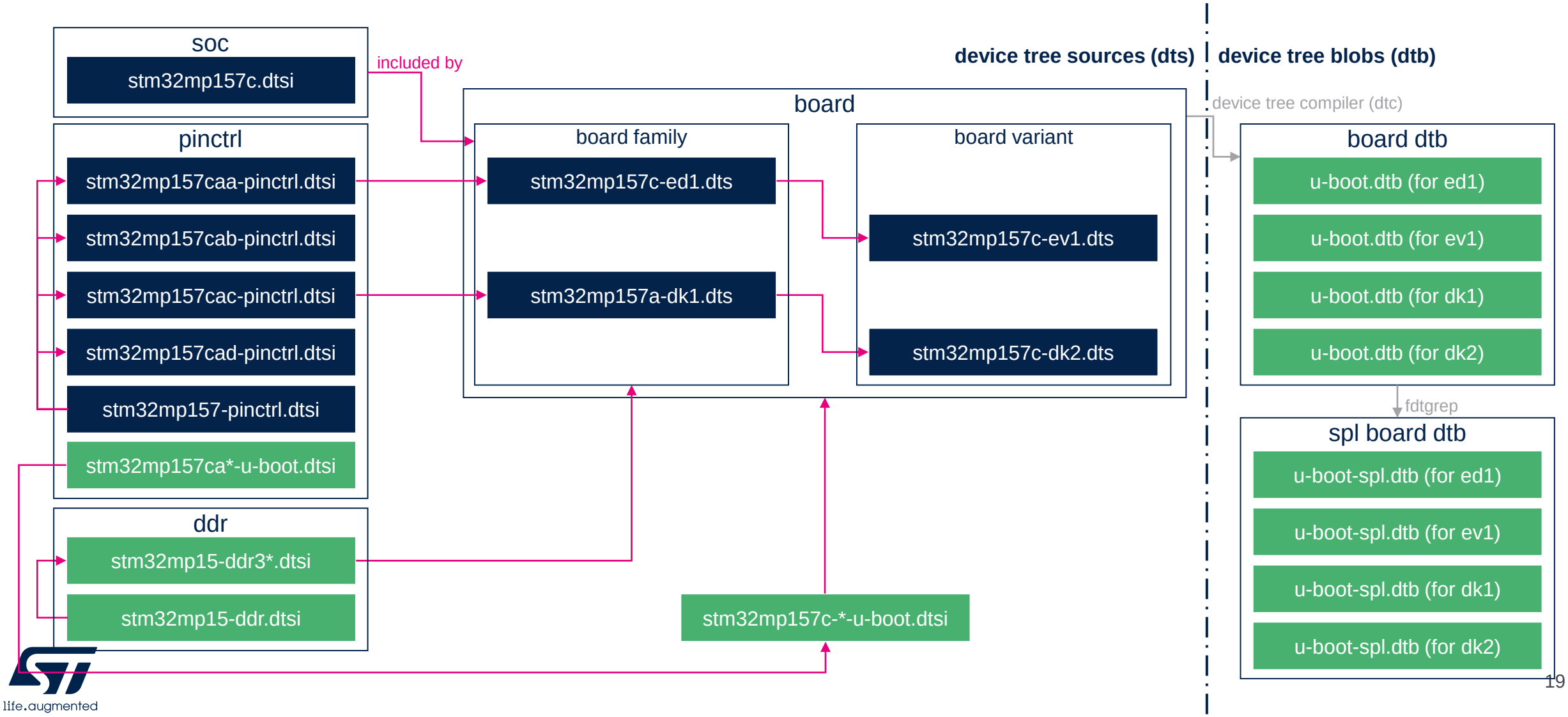
life.augmented

- **ed: evaluation daughter board** (embedding the STM32MP1)
- **ev: evaluation board** (ed plugged on the mother board)
- **dk: discovery kit**

Legend

Upstreamed device tree

U-Boot, overloading copies of Linux files



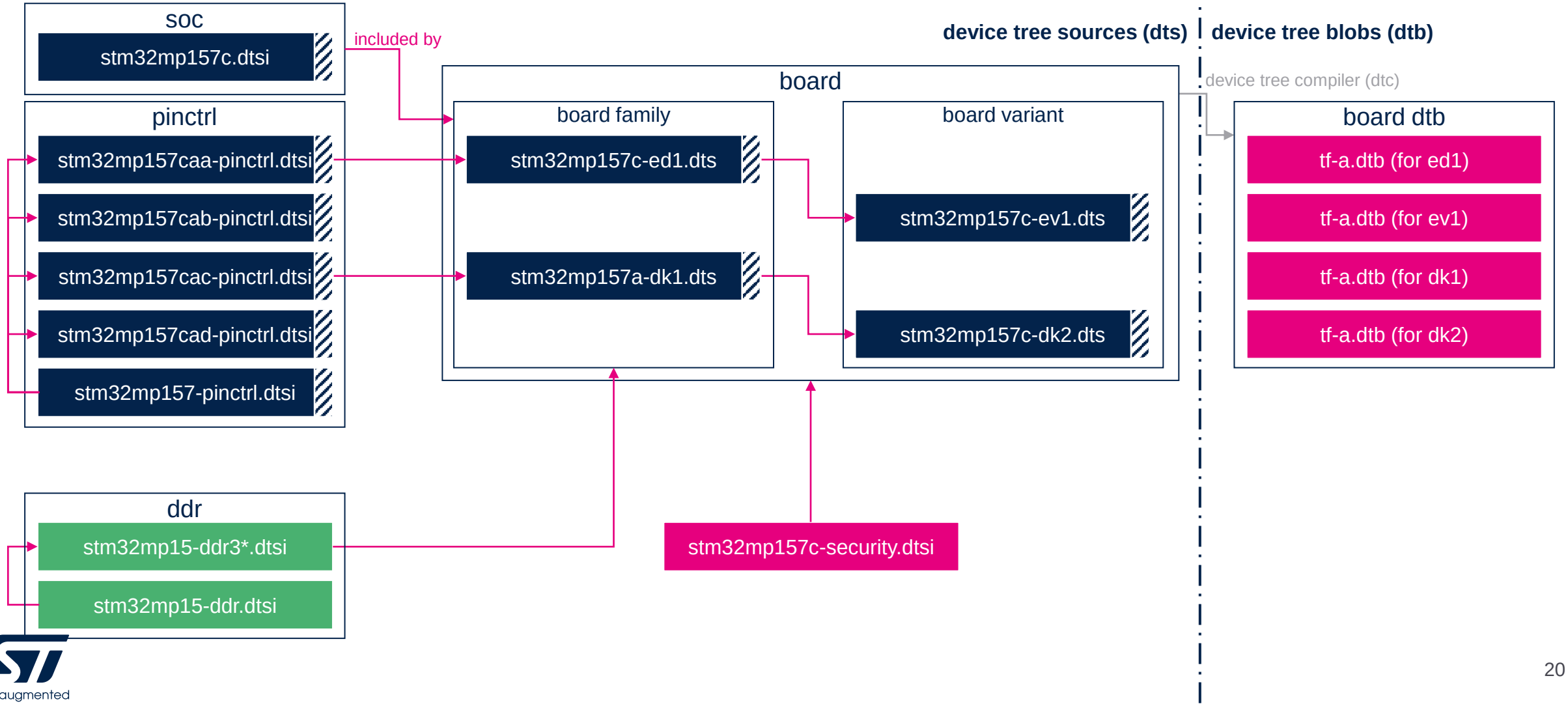
life.augmented

- **ed: evaluation daughter board** (embedding the STM32MP1)
- **ev: evaluation board** (ed plugged on the mother board)
- **dk: discovery kit**

Legend

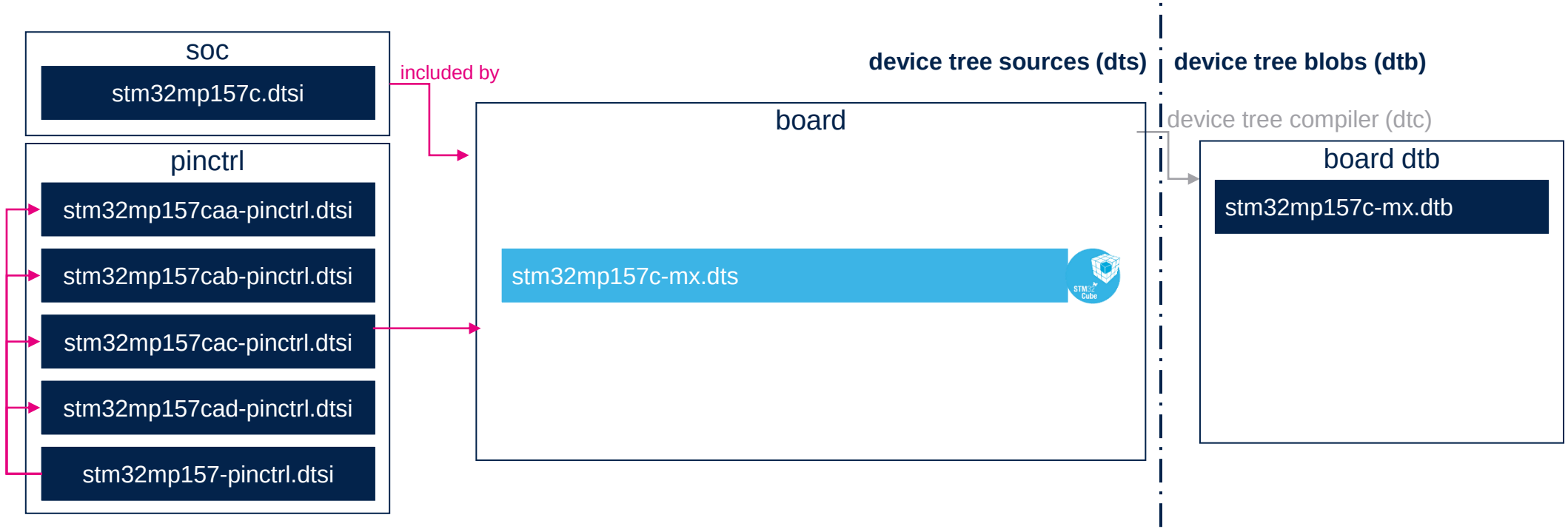
Upstreamed device tree

TF-A, overloading subsets of Linux files copies and using DDR config from U-Boot



life.augmented

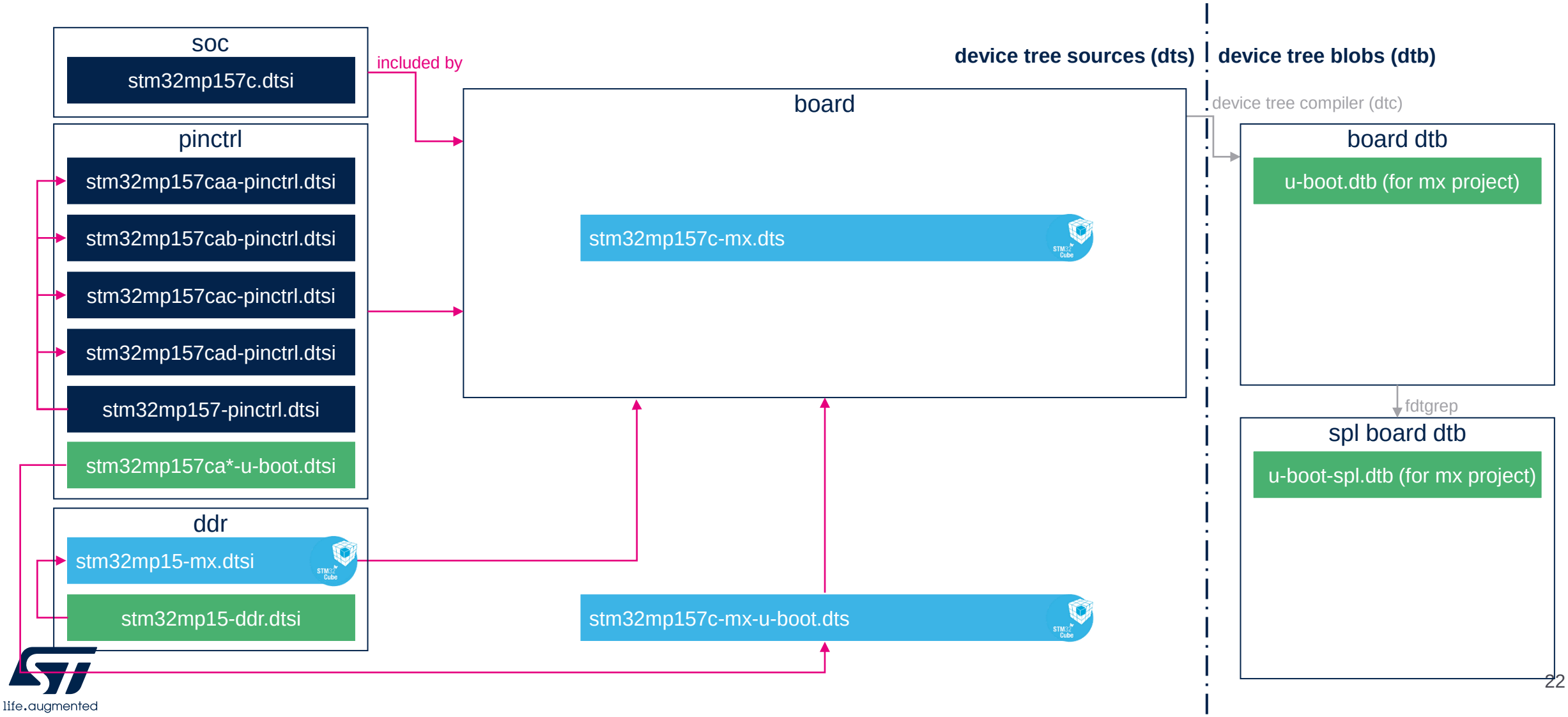
Stm32cubemx generation



STM32CubeMX Device tree generation for Linux

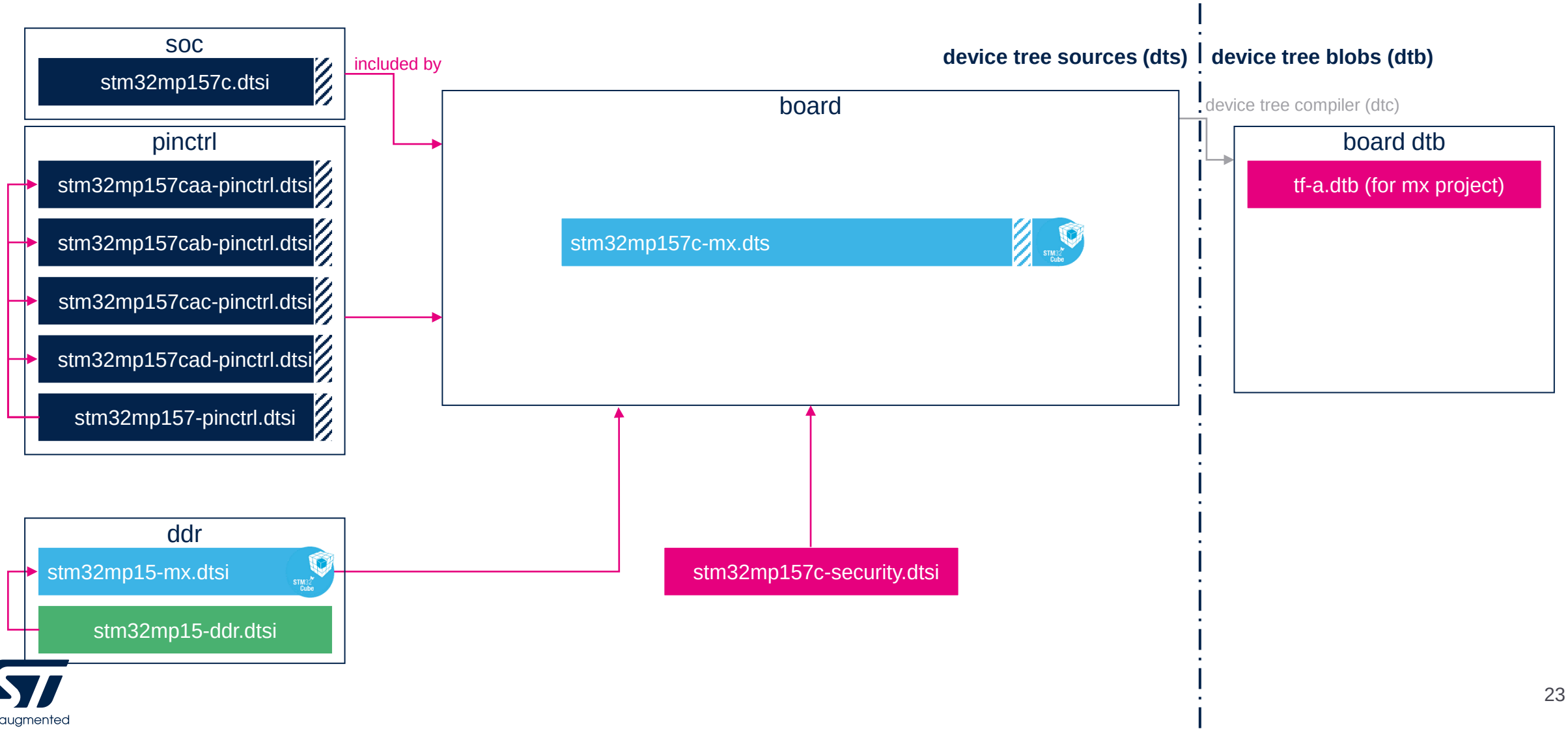
Stm32cubemx generation

U-Boot, overloading copies of Linux files



Stm32cubemx generation

TF-A, overloading subsets of Linux files copies and using DDR config from U-Boot



Openstlinux flash memory mapping

Flash partitions (minimal)

Size	Component	Comment
Remaining area	userfs	The user file system contains user data and examples
768MB	rootfs	Linux root file system contains all user space binaries (executable, libraries, …) and kernel modules
16MB	vendorfs	This partition is preferred to the rootfs to put third parties proprietary binaries and ensure that they are not contaminated by any open source licence, such as GPL v3
64MB	bootfs	The boot file system contains: - (option) the init ram file system, that can be copied to the external RAM and used by Linux before mounting a fatter rootfs - Linux kernel device tree (can be in a Flattened Image Tree - FIT) - Linux kernel U-Boot image (can be in a Flattened Image Tree - FIT) - For all flashes but the NOR: the boot loader splash screen image, displayed by U-Boot - U-Boot distro config file extlinux.conf (can be in a Flattened Image Tree – FIT)
2MB	ssbl	The Second Stage Boot Loader (SSBL) is U-Boot, with its device tree blob (dtb) appended at the end
256kB to 512kB (*)	fsbl	The First Stage Boot Loader is ARM Trusted Firmware (TF-A) or U-Boot Secondary Program Loader (SPL), with its device tree blob (dtb) appended at the end. At least two copies are embedded. Note: due to ROM code RAM needs, FSBL payload is limited to 247kB.

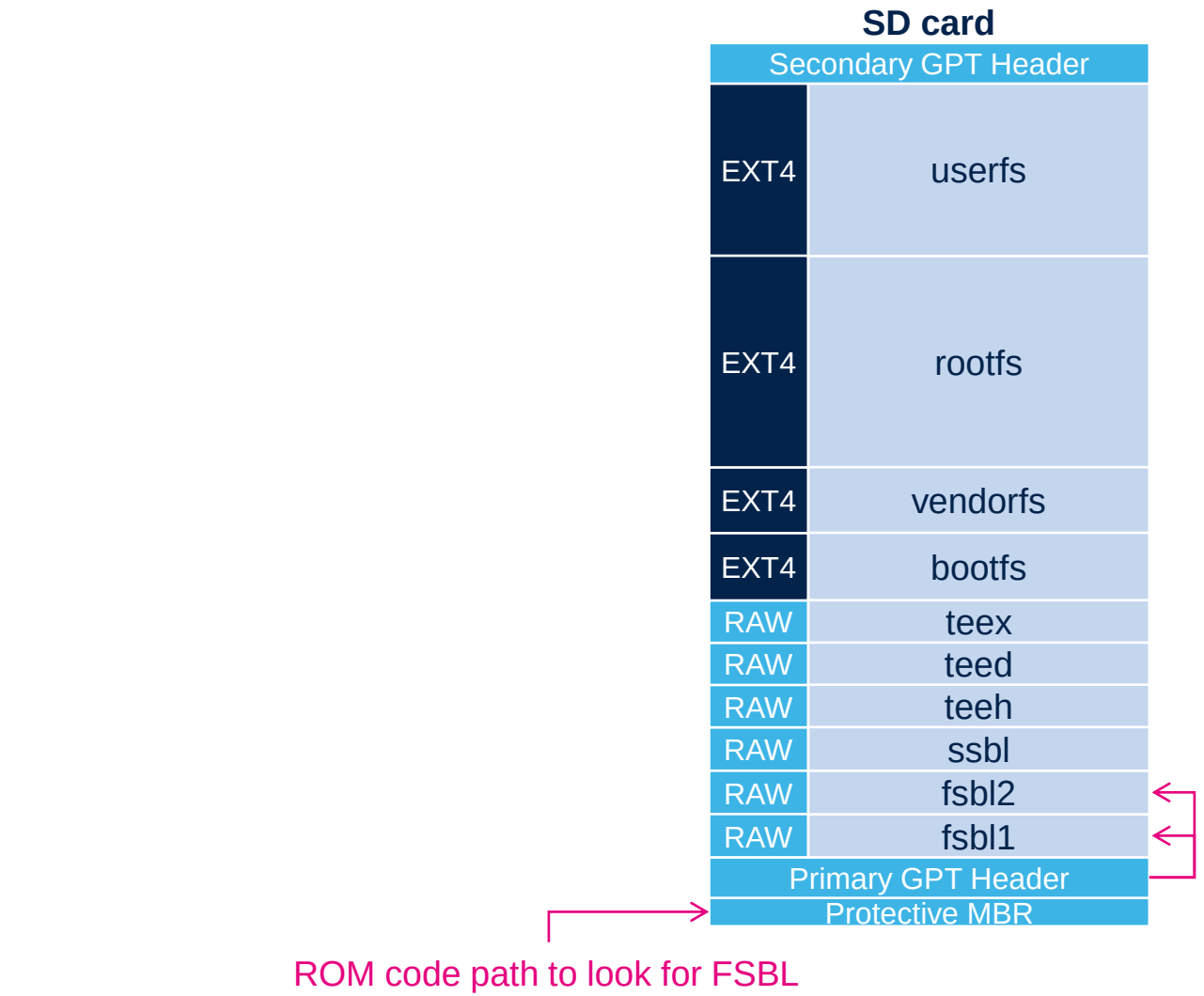
(*): the partition size depends on the flash technology, to be aligned on block erase size for NOR (256kB) / NAND (512kB)

Flash partitions (optional)

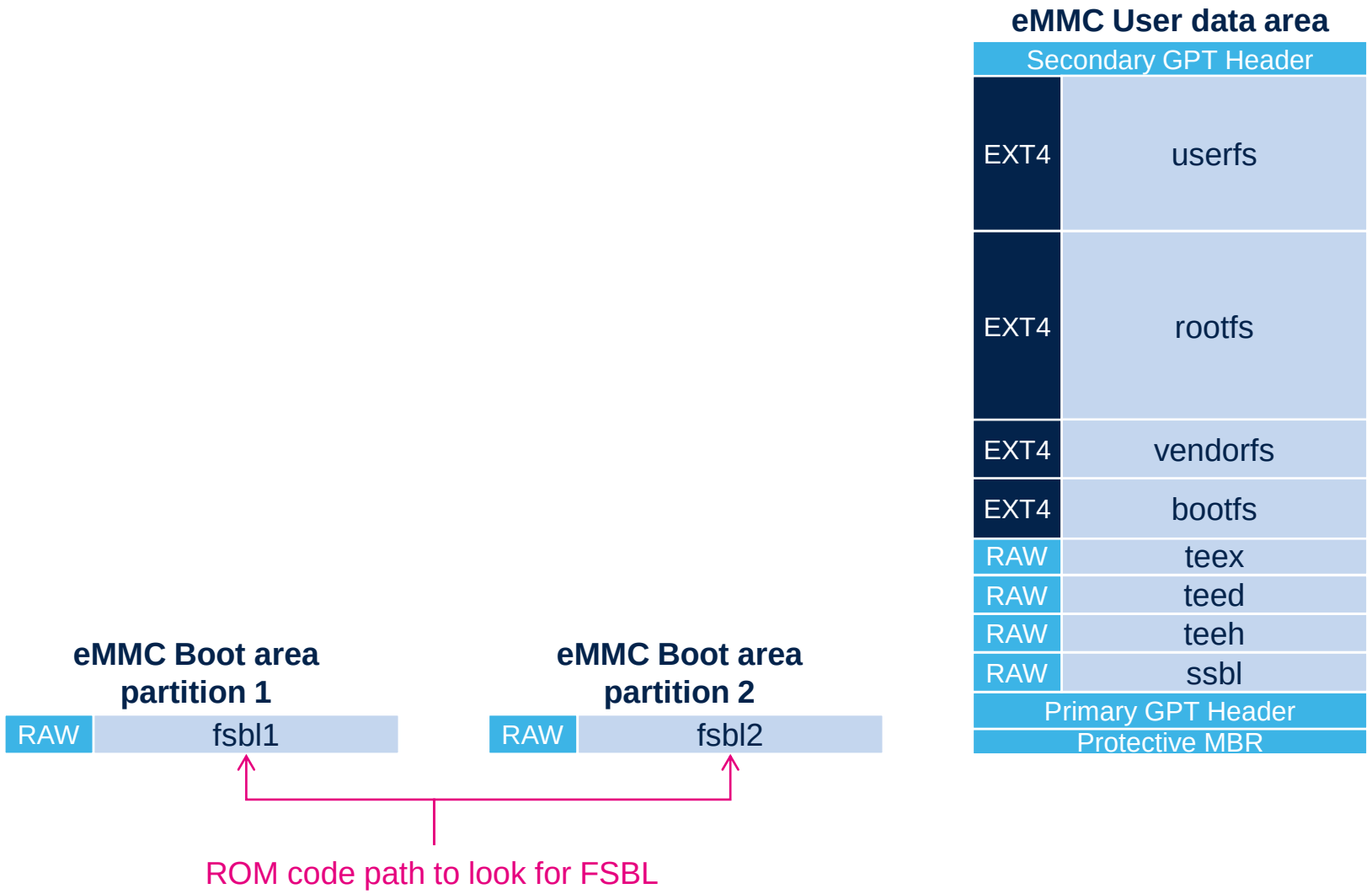
Size	Component	Comment
256kB (*)	logo	This partition contains the boot loader splash screen image while booting on NOR flash (for all other flashes, the image is stored in the bootfs partition)
256kB to 512kB (*)	teeh	OP-TEE header
256kB to 512kB (*)	teed	OP-TEE pageable code and data
256kB to 512kB (*)	teex	OP-TEE pager

(*): the partition size depends on the flash technology, to be aligned on block erase size for NOR (256kB) / NAND (512kB)

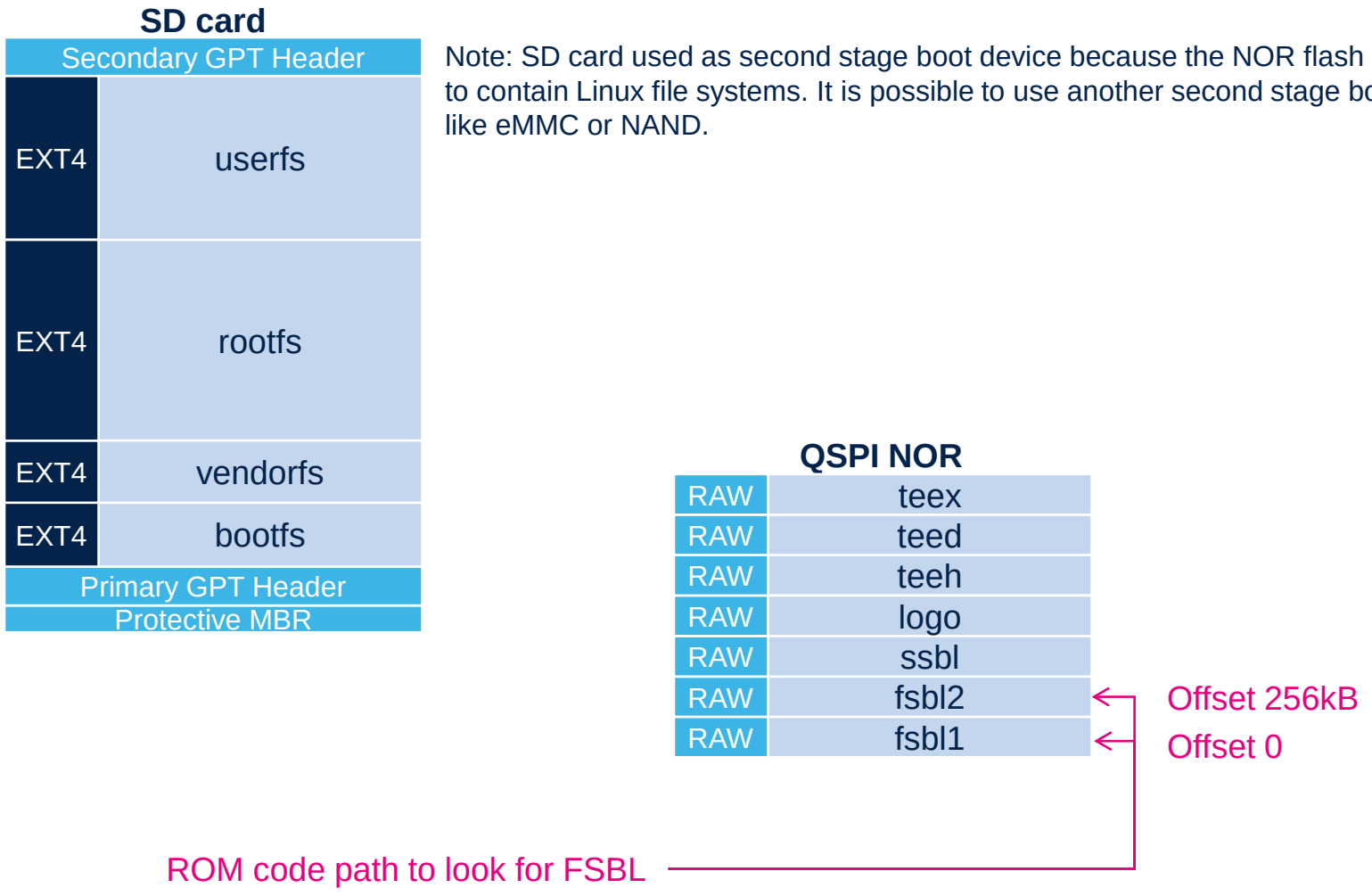
SD card memory mapping



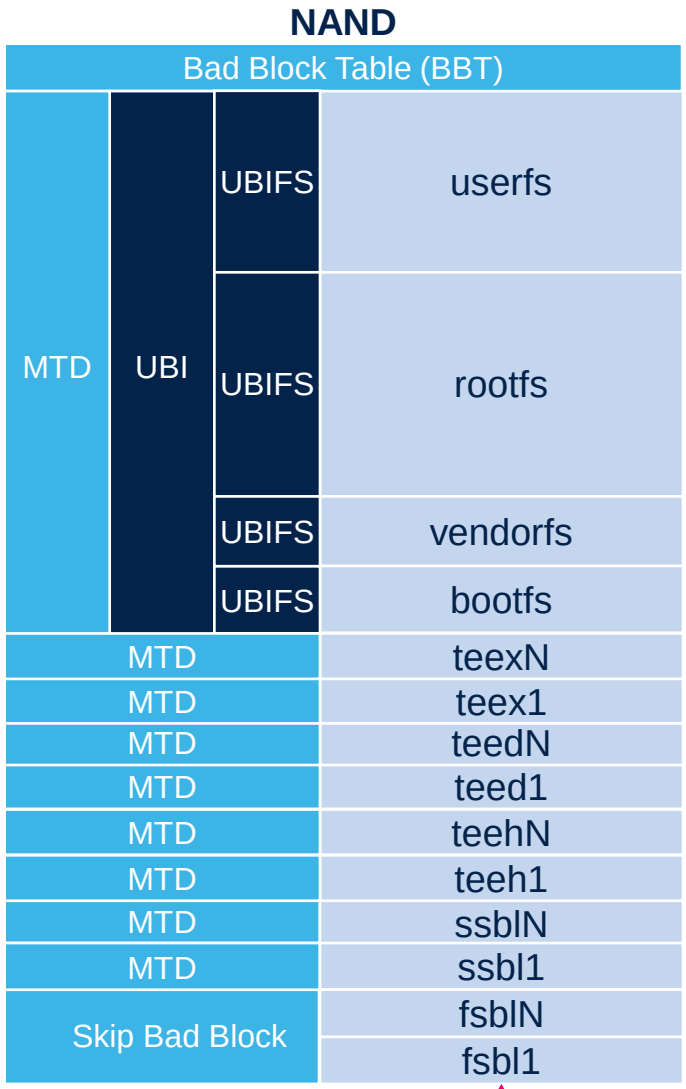
Emmc memory mapping



NOR memory mapping



NAND memory mapping



Note: SSBL and OP-TEE partitions may move to UBI format when FSBL supports it

Note: in the Skip Bad Block area, the number of copies and the margin have to defined in STM32CubeProgrammer flash layout, depending on the product expected life time and firmware update strategy

life.augmented

Appendix

Authentication