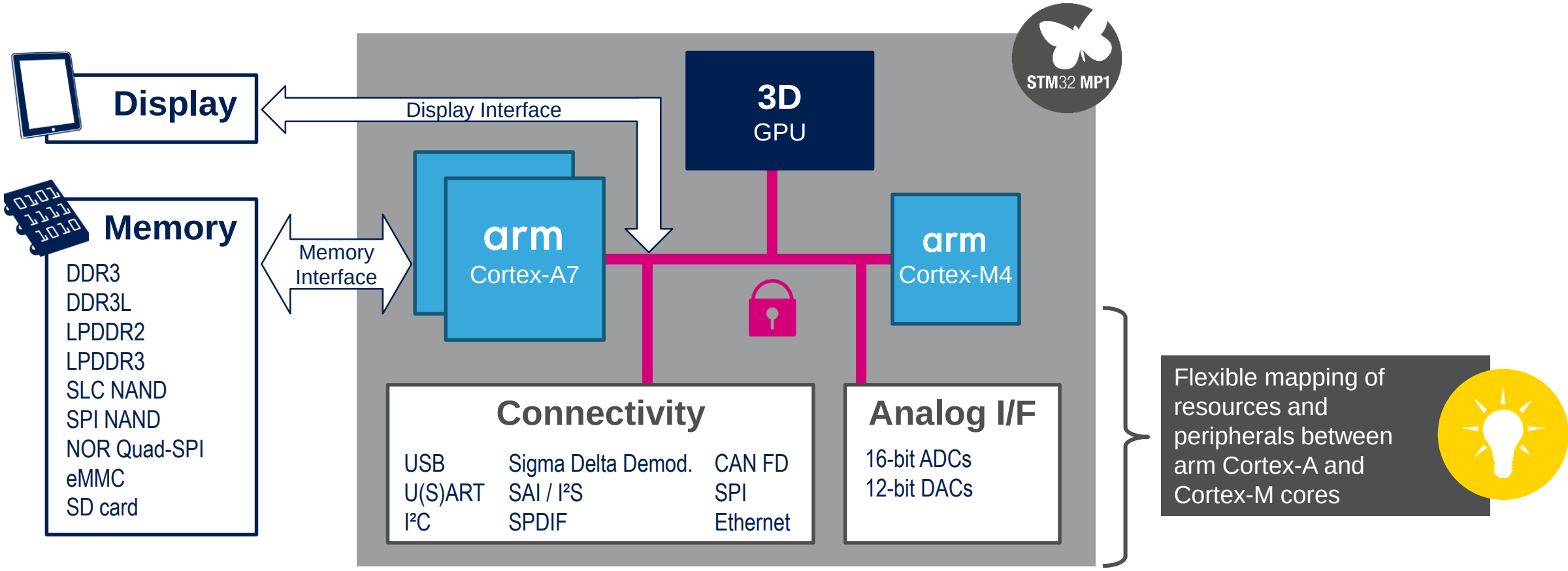
STM32MP1 MPU series

New microprocessor series with enhanced performance thanks to its multicore architecture and graphical processor

life.augmented

Rich Feature Set

Advanced & Flexible Architecture with 3D GPU



life.augmented

STM32MP1: A General Purpose MPU

Suitable for all Developer Types and Multiple Applications

Developer profile

MCU users new to MPU — MCU → MPU

Mixed MCU and MPU users — MCU + MPU

Pure MPU users — MPU

STM32 MP1

Possible applications

Industrial

Consumer

Health & Wellness

Home

life.augmented

Supported by the STM32 Ecosystem

All the Tools for Successful MPU Development

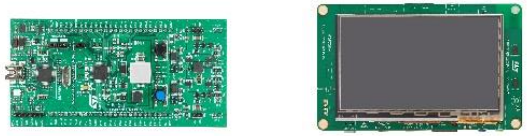
Software

STM32 Cube

STM32 CubeMX

STM32 CubeProgrammer

Hardware



Discovery boards

Evaluation boards

Customer support

FAE - Worldwide Customer Support

community.st.com

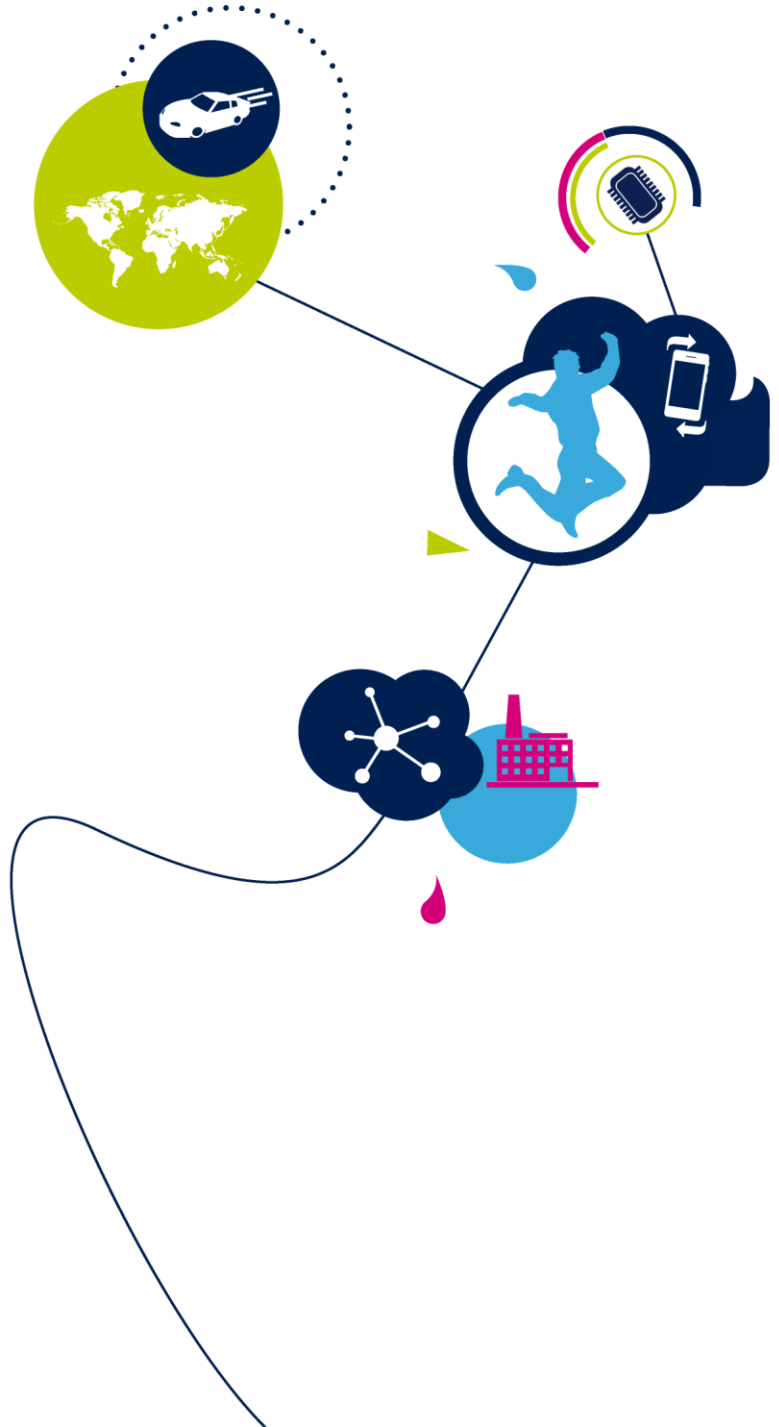
Partner Program

life.augmented

STM32MP1 – HW ARCH

Hardware Architecture

Revision 1.0



life.augmented

STM32MP1 Block Diagram

STM32MP157

System
- 5x LDOs
- Crystal & Internal oscillators
- MDMA + 2x DMA
- Reset and Clock
- Watchdogs (2x I & W)
- 96-bit unique ID
- Up to 176 GPIOs

Security
- TrustZone
- DES, TDES, AES-256
- SHA-256, MD5, HMAC
- 3x Tamper Pins with 1 active
- T°, V and 32KHz detection
- Secure ROM and RAMs
- Secure Peripherals
- Secure RTC
- Analog true RNG

Dual Cortex-A7 @ 650MHz
- Core 1 @ 650MHz L1 32KB I / 32KB D
- Core 2 @ 650MHz L1 32KB I / 32KB D
- NEON SIMD
- NEON SIMD
- 256KB L2 cache

Cortex-M4 @ 209MHz
- FPU
- MPU

DDR3/DDR3L 32-bit @ 533MHz
LPDDR2/LPDDR3 32-bit @ 400MHz

- System RAM 256KB
- MCU System RAM 384KB
- Retention RAM 64KB
- Backup RAM 4KB
- Boot ROM 128KB
- OTP Fuse 3Kb

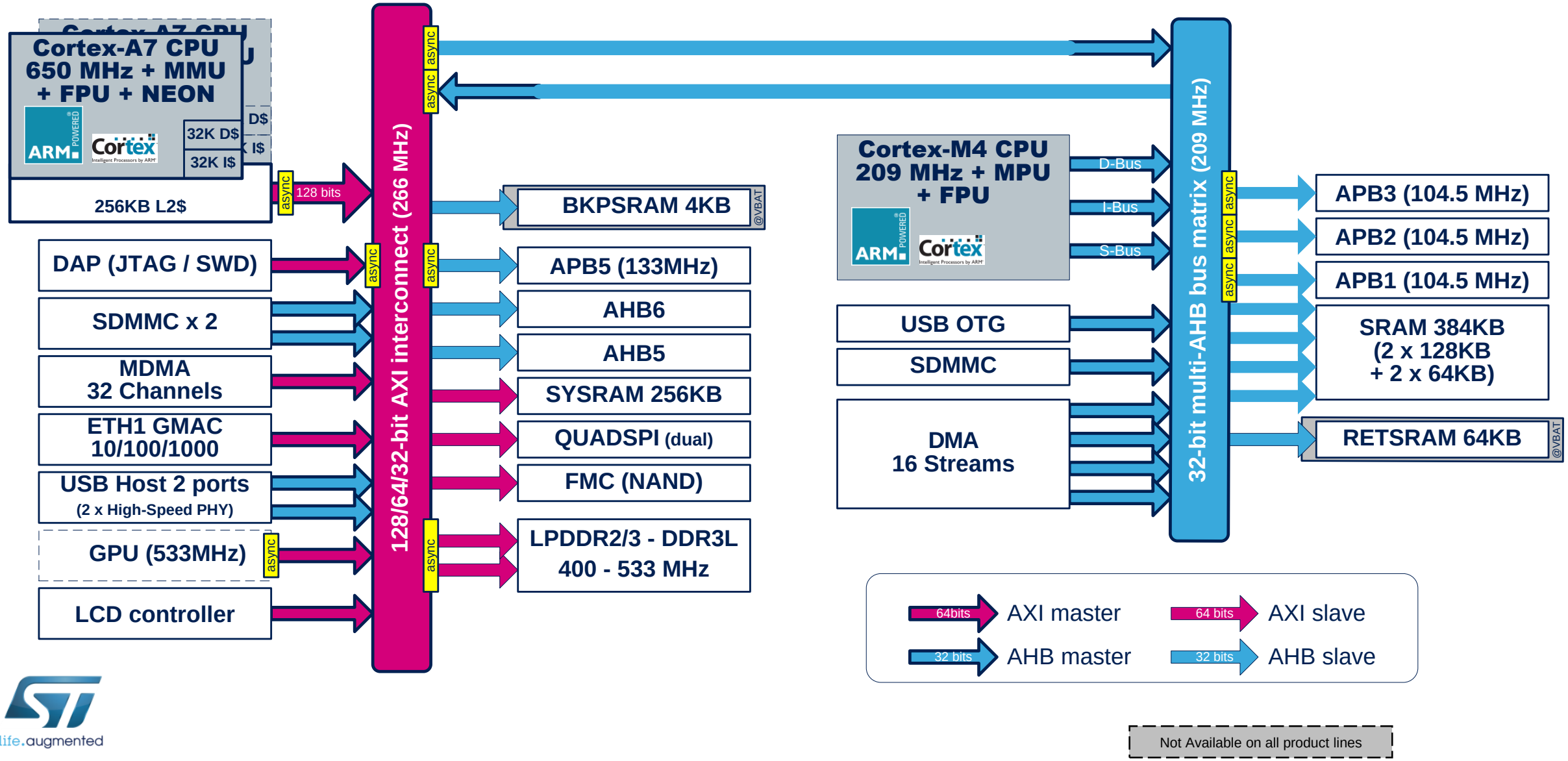
Control
- 2x 16-bit motor control PWM synchronized AC timer
- 10x 16-bit timer
- 5x 16-bit LP timer
- 2x 32-bit timer

3D GPU OpenGL ES2.0 @ 533MHz
26Mtri/sec, 133Mpix/sec

Connectivity
- 24-bit Parallel RGB Display
- MIPI DSI 2 lanes @ 1Gbps
- Camera Interface
- HDMI CEC
- 1Gbps Ethernet
- 2x FDCAN / TTCAN
- 2x USB2.0 Host HS
- USB2.0 OTG FS/HS
- MDIO
- DFSDM 8 channels / 6 filters
- 6x I²C
- 4x UART, 4x USART
- 6x SPI / 3x I²S
- 4x SAI
- SPDIF Tx / Rx 4 inputs
- Dual Quad SPI
- 3x SDIO3.0 / SD3 / eMMC 4.51
- 16-bit SLC NAND, 8-bit-ECC

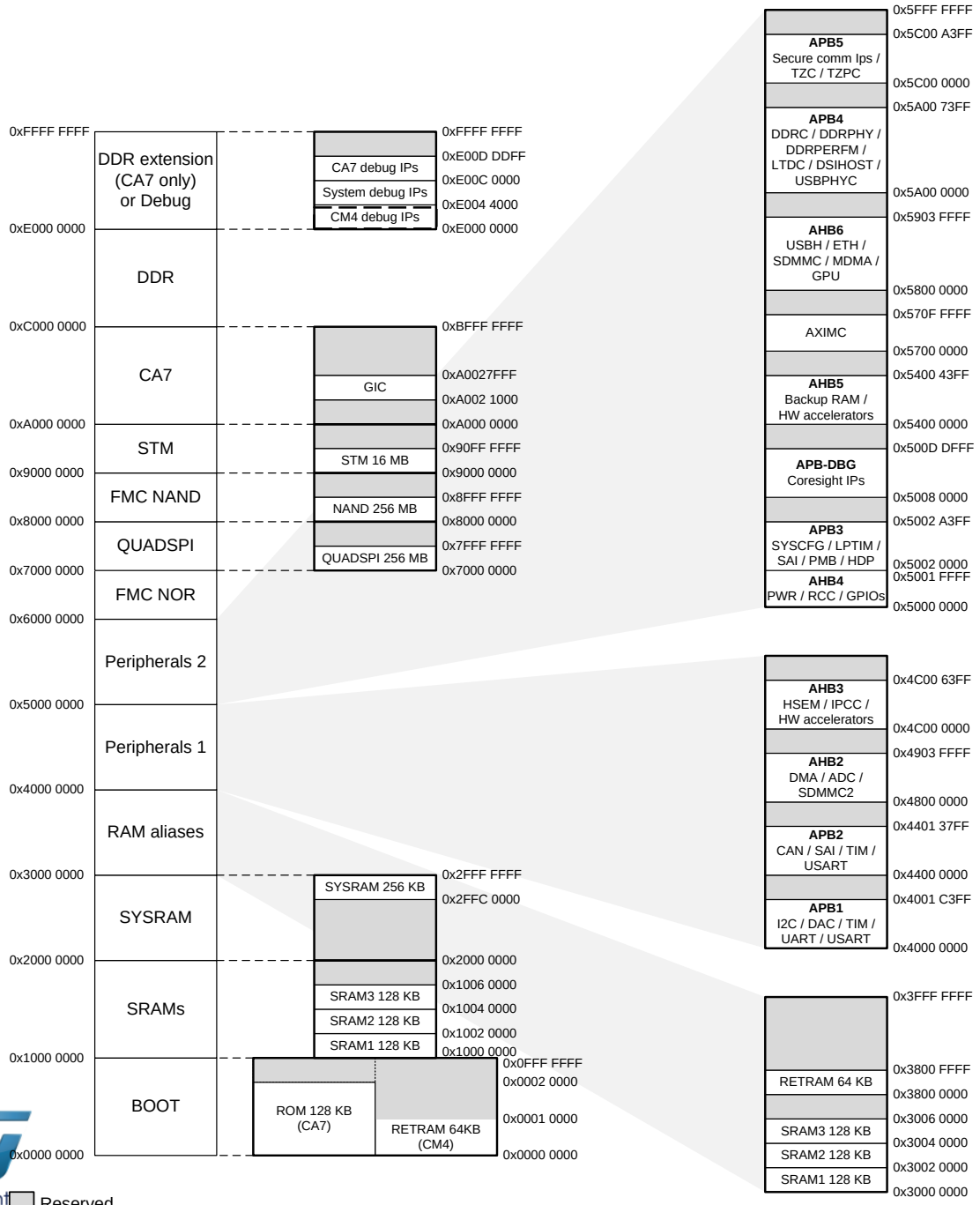
Analog
- 2x 16-bit ADC 3MSPS 22 channels
- 2x 12-bit DAC
- Temperature sensor

STM32MP1 bus architecture overview



life.augmented

Memory Map overview



- Uniform memory map
  - All peripherals visible at same address of every masters
  - No Remap

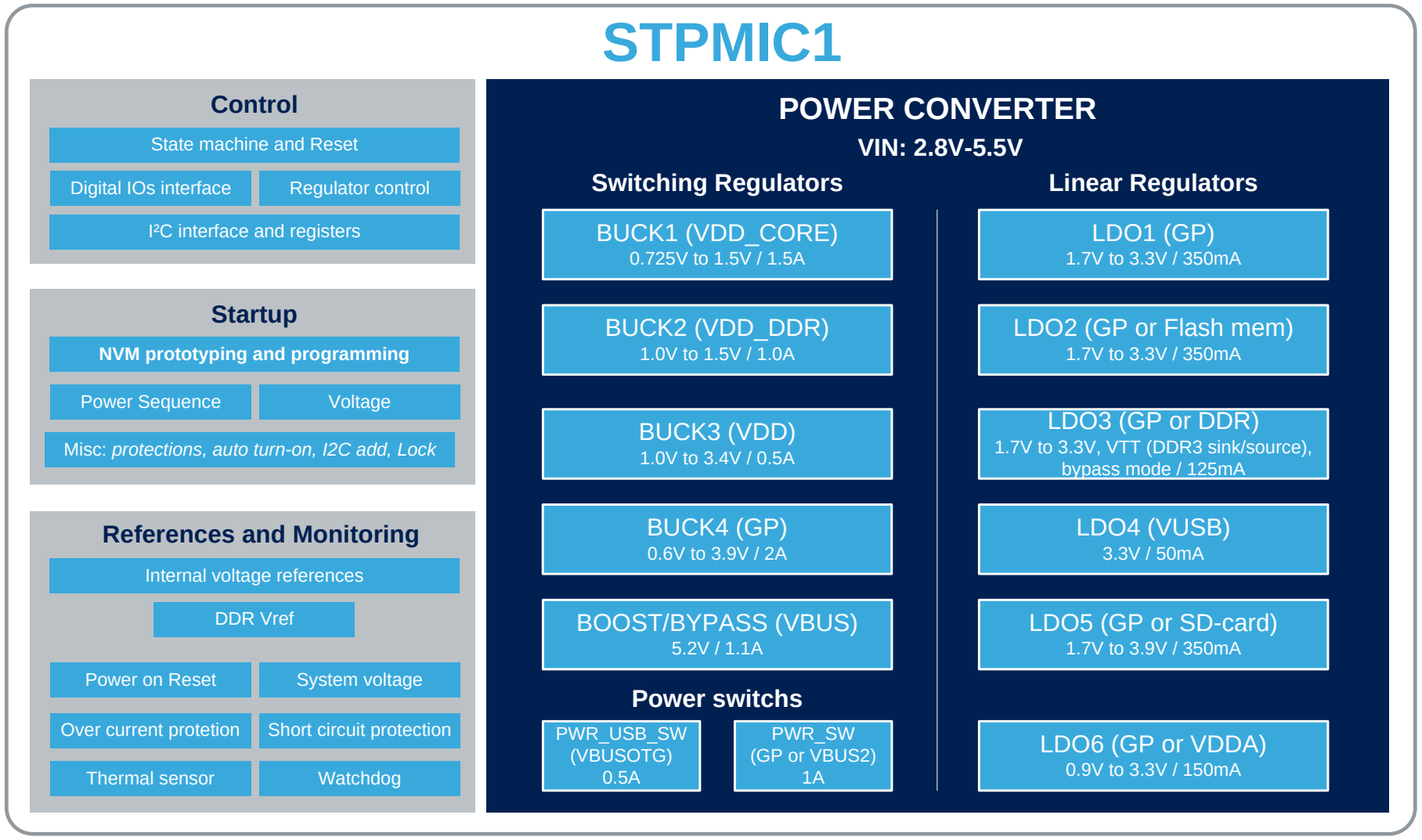
Power Supplies

Name	Typical or Range	Description
VDD	**1.7V – 3.6V**	Power supply input for I/Os
VDD_ANA		Power supply input for system analog like RCC, PWR. To be connected to VDD
*VDD_DSI***		Power supply input for DSI regulator. To be connected to VDD
VDD_PLL		Power supply input for PLLs. To be connected to VDD
VDD_CORE	**1.2V**	Power supply input for Digital Core domain
VDDA	**1.7V – 3.6V**	Analog Power supply input for ADCs, DACs and voltage reference buffers
VDDQ_DDR	**1.2V / 1.35V / 1.5V***	Power supply input for DDR Physical Interface (PHY) and IOs
VDD3V3_USBHS/FS	**3.3V**	Power supply input for USB Physical Interface (PHY) and IOs
Internally generated Power Supplies		
VDDA1V8_REG	**1.8V**	Analog Power Supply input or output, used internally for USB Physical Interface (PHY)
*VDDA1V8_DSI***		Analog Power supply input for DSI Physical Interface (PHY), to be connected to VDDA1V8_REG
VDDA1V2_DSI_REG**	**1.2V**	Analog Power supply output, used internally for DSI PLL
*VDDA1V2_DSI_PHY***		Analog Power supply input for DSI Physical Interface (PHY), to be connected to VDDA1V2_DSI_REG
VDDA1V1_REG	**1.1V**	Analog Power supply output for USB Physical Interface (PHY)

life.augmented

(*) LPDDR2/LPDDR3, DDR3L and DDR3 respectively
(**) only on STM32MP157 line

STPMIC1 Block Diagram



Package: WFQFN 5x6x0.8 44L pitch 0.4mm

life.augmented

Peripherals (1)

Timers	Advanced	16 bits	2
	General purpose	16 bits	8
		32 bits	2
	Basic	16 bits	2
	LP Timer	16 bits	5
	M4 Systick	24 bits	1
	A7 Timers	64 bits	2 x 4 (Secure, Non-Secure, Virtual, Hypervisor)
	RTC/AWU		1 (Securable)
	Watchdog		3 (Independent, Independent Secure, Window)
Communication Peripherals	SPI		6 (1 Securable)
		having I2S	3
	I2C (with SMB/PMB support)		6 (2 securable)
	USART (Smartcard, SPI, IrDA, LIN) + UART (IrDA, LIN)		4 + 4 (including 1 Securable USART)
	SAI		4 (up to 8 audio channels), with I^2S master/slave, PCM input, SPDIF-TX
	USB	EHCI/OHCI Host	2 ports
			embedded HS PHY with BCD
		OTG HS	Yes, Embedded FS or HS PHY with BCD
		Embedded PHYs	3 (2 x High-Speed + 1 x Full-Speed)
	SPDIFRX		4 inputs
	FDCAN		2 (1 x TT-FDCAN), Clock Calibration, 10 Kbytes shared buffer
	CEC (HDMI Consumer Electronics Control)		1
	Including the following securable		1 x USART, 1 x SPI, 2 x I2C on Securable GPIOs

Boot Source

Boot Source

- All Peripherals Input/Output (when existing) are mapped through GPIO alternate functions

Not Available on all product lines or packages

life.augmented

Peripherals (2)

SDMMC (SD, SDIO, eMMC)		3 (8 + 8 + 4 bits), eMMC or SD
QuadSPI		Yes (dual-quad)
FMC	Parallel Address/Data 8/16-bits	4 x CS, up to 4 x 64MBytes
	Parallel AD-Mux 8/16-bits	4 x CS, up to 4 x 64MBytes
	NAND 8/16-bits	Yes, 1 x CS, SLC, BCH4/8
Gigabit Ethernet		MII, RMII, GMII, RGMII with PTP and EEE
DMA		3 instances (1 Securable) , 48 physical channels in total
Cryptography		DES, TDES, AES-256, dual instances (Secure and non-Secure)
Hash		SHA-256, MD5, HMAC, dual instances (Secure and non-Secure)
Random number generator		True-RNG, dual instances (Secure and non-secure)
Fuses (One Time Programming)		3072 effective bits (Secure, >1500 bits available to user)
Digital Camera Interface	bus width	8, 10, 12 or 14-bits
GPIOs with interrupt (total count)		98, 148 or 176 depending on package
	Securable GPIOs	8
	Wakeup pins	4 or 6 depending on package
	Tamper pins (Active tamper)	2 or 3 (1) depending on package
DFSDM		8 input channels with 6 filters
16 bit synchronized ADC		2
	ADC channels in total	17 or 22 depending on package
12 bit DAC		2
Internal ADC/DAC VREF generation		1.5V, 1.8V, 2.048V, 2.5V or VREF+ input
	VREF+ input pin	Yes

Boot Source

Boot Source

Boot Source

Not Available on all product lines or packages

life.augmented

Graphic and Display

- LCD-TFT controller, up to 24-bit // RGB888
  - up to WXGA (1366 × 768) @60 fps
  - Two layers with programmable colour LUT
- 3D GPU: Vivante® - OpenGL® ES 2.0
  - 533 MHz, up to 26 Mtriangle/s, 133 Mpixel/s
- MIPI® DSI 2 data lanes up to 1 GHz each
  - up to WXGA (1366 × 768) @60 fps

Not Available on all product lines or packages

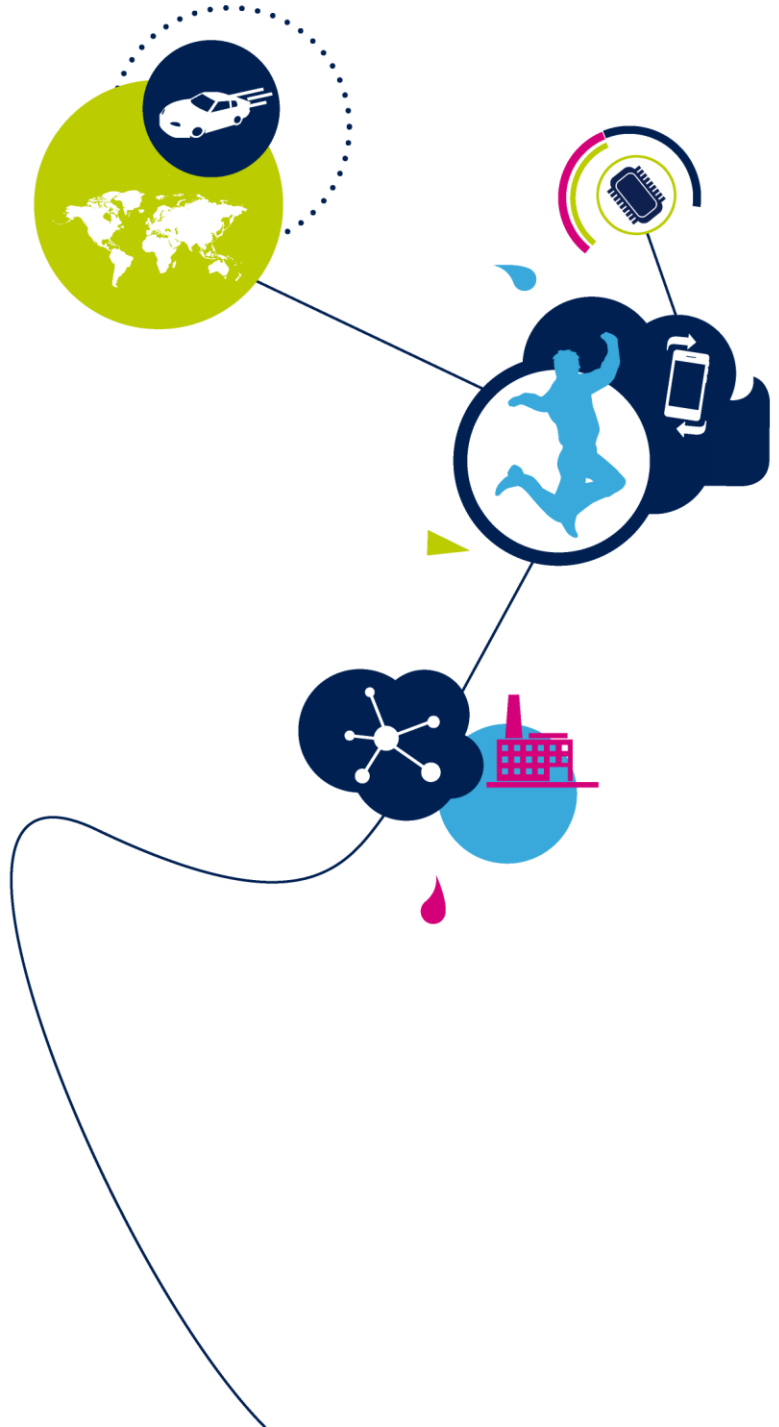
OTP Fuses

- OTP Fuses are One Time Programming memory
  - Initial bits are ‘0’ and are irreversibly programmed to ‘1’
  - Incremental programing of bits in a 32-bit word is possible
- Handled thru BSEC controller IP
  - Programming, reading, status and locking handled by BSEC
  - Lock mechanism to avoid read and/or program (32-bits granularity)
- OTP Content
  - Product configuration and Trimming values set by ST during production
  - Secrets and unique identification numbers set by ST during production
  - Device configuration set by OEM (e.g. MAC address, boot source, security mode, etc...)
  - Secrets set by OEM (e.g. for secure boot)
  - Up to 1184 bits available for other OEM purposes

STM32MP1 SWARCH

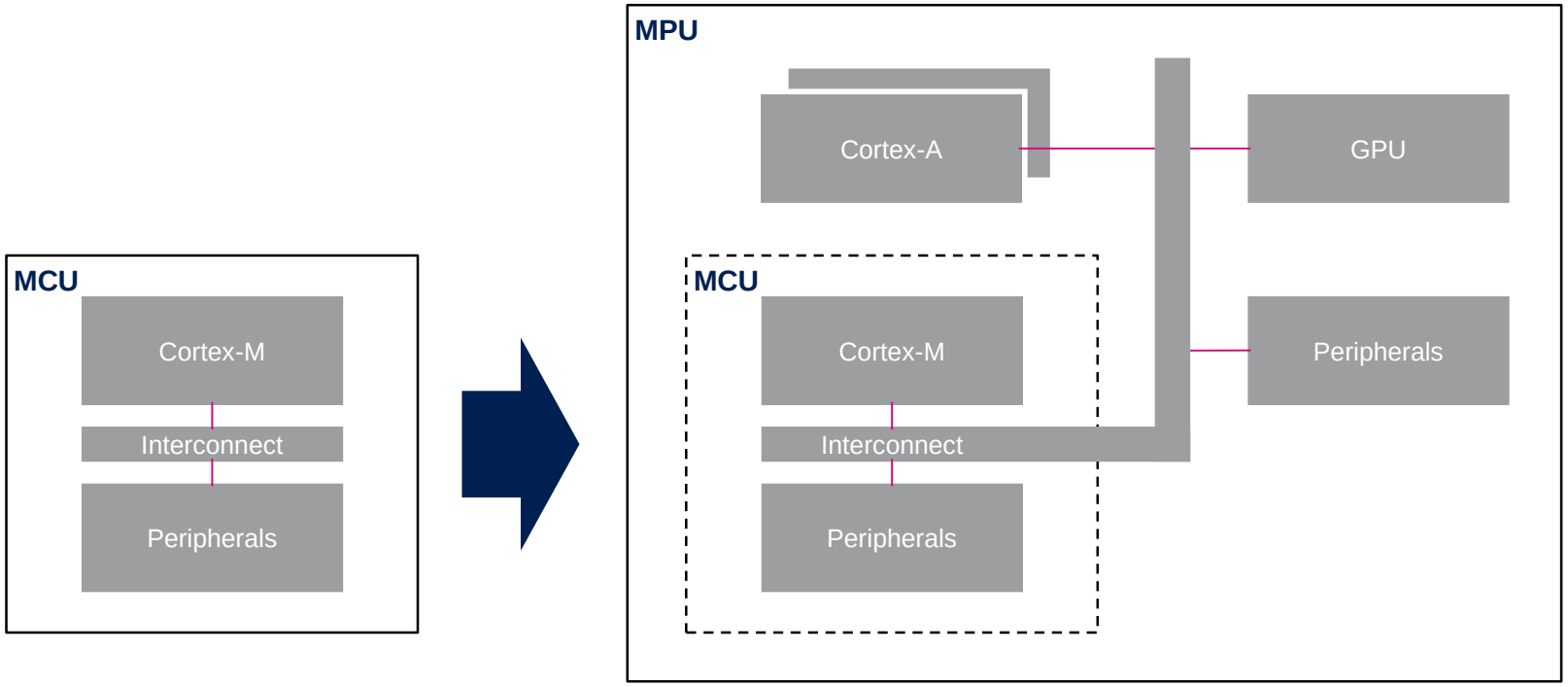
Embedded Software architecture

Revision 1.0



life.augmented

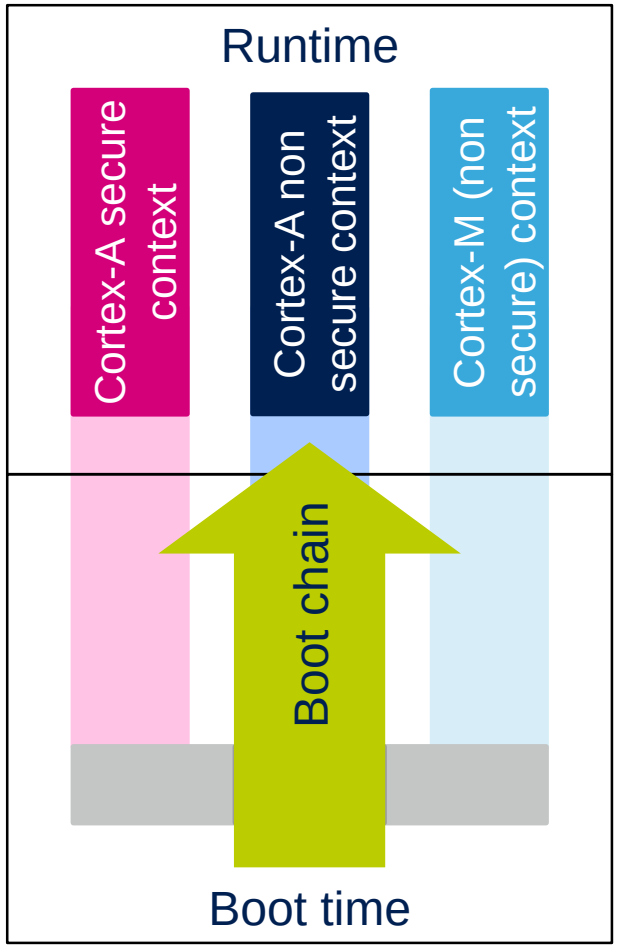
From MCU to MPU



Source: ST Wiki article [Getting started with STM32 MPU devices]

Multiple-core architecture concepts

- Hardware execution context
  - « a core and a security mode »
- Firmwares executed runtime contexts
  - Arm Cortex-A secure (Trustzone) executes OP-TEE
  - Arm Cortex-A non secure executes Linux
  - Arm Cortex-M (non secure) executes STM32Cube
- Peripheral assignment to the runtime contexts
  - Assigned or shared

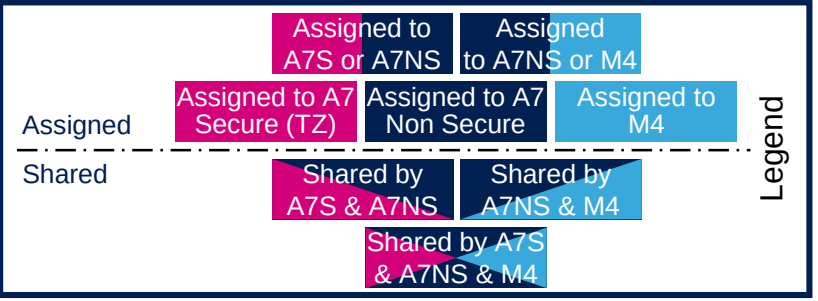


life.augmented

Source: ST Wiki article [Getting started with STM32 MPU devices]

Peripherals sharing

Source: ST Wiki article [STM32MP1 peripherals overview]



Coprocessor
IPCC

Cortex-M4
(co-processor)

Core
STGEN
SYSCFG
RTC

Interrupts
EXTI
GIC
NVIC

Watchdog
IWDG1
IWDG2
WWDG

DMA
DMAMUX
DMA1
DMA2
MDMA

RAM
SYSRAM (256KB)
DDR
BKPSRAM (4KB)
MCU SRAM12 (128KB)
MCU SRAM34 (128KB)
RETRAM (64KB)

Timers
TIM (x2)
TIM (x12)
LPTIM (x5)

IOs
GPIOA-K
GPIOZ

Security
RNG2
HASH2
ETZPC
CRYP2
CRC2
TZC
RNG1
HASH1
TAMP
CRYP1
CRC1

Visual
GPU
DSI
LTDC
DCMI
CEC

Power and thermal
RCC
PWR
DTS

Trace and debug
DBGMCU
HDP
STM

Persistent storage
BSEC
QUADSPI
FMC
SDMMC1/2 (*)
Mass storage

Networking
FDCAN
ETH

Low speed interfaces
USART (x3)
UART (x4)
SPI6
USART1
I2C (x4)
SPI (x2)
I2C4
I2C6

(*): some peripherals are multi-functions and may be mapped differently by the customer

Analog
VREFBUF
DAC
DFSDM (*)
ADC

SDMMC3 (*)
USBH (Host)
OTG (USB OTG)
High speed interfaces

SPI I2S (x3) (*)
SPDIFRX
SAI (x4)
Audio

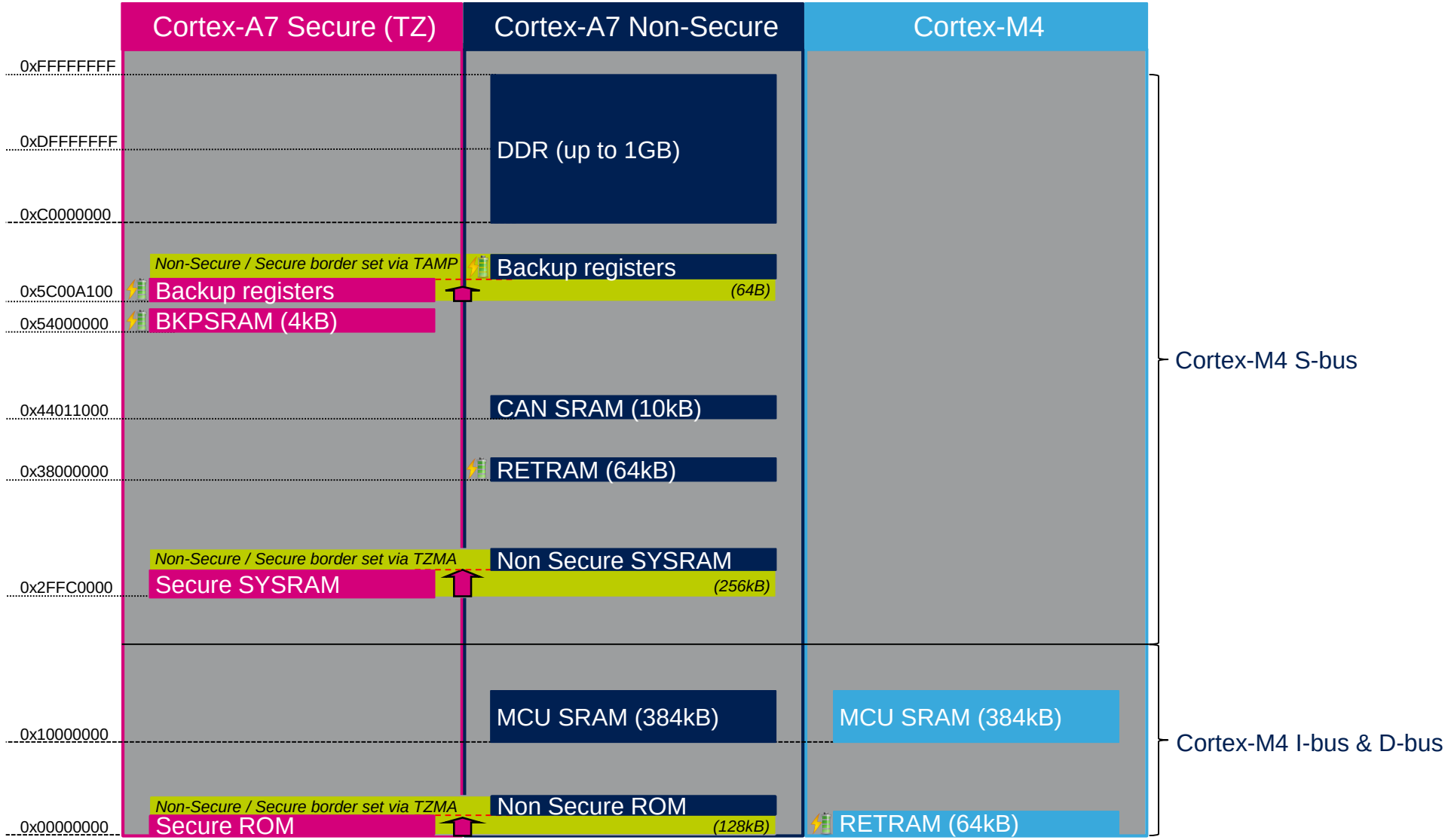
life.augmented

Legend

: memory content preserved during standby

Software memory mapping

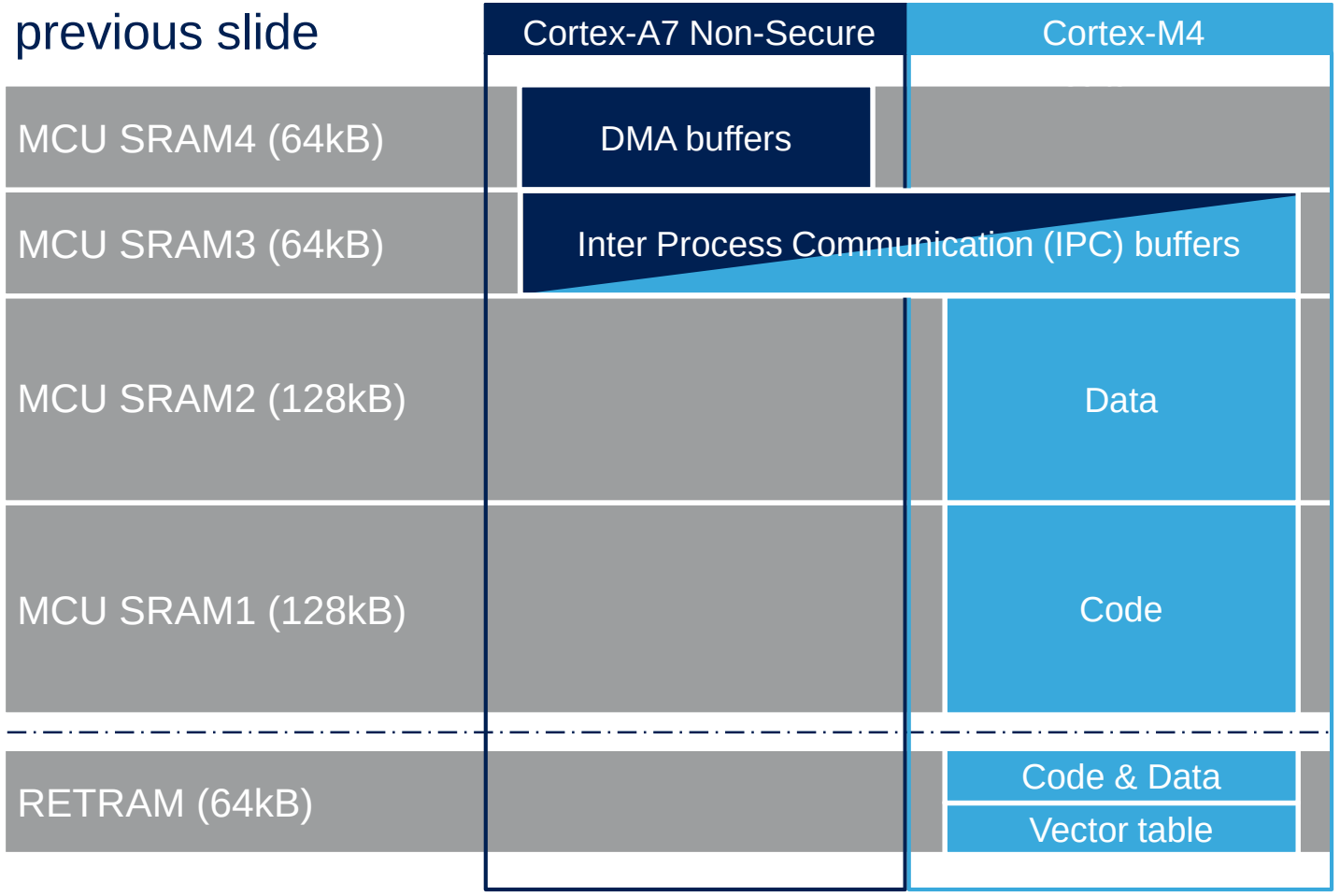
- The memory mapping below is a subset of all regions that are really exposed at hardware level.



life.augmented

Shared RAM memory mapping

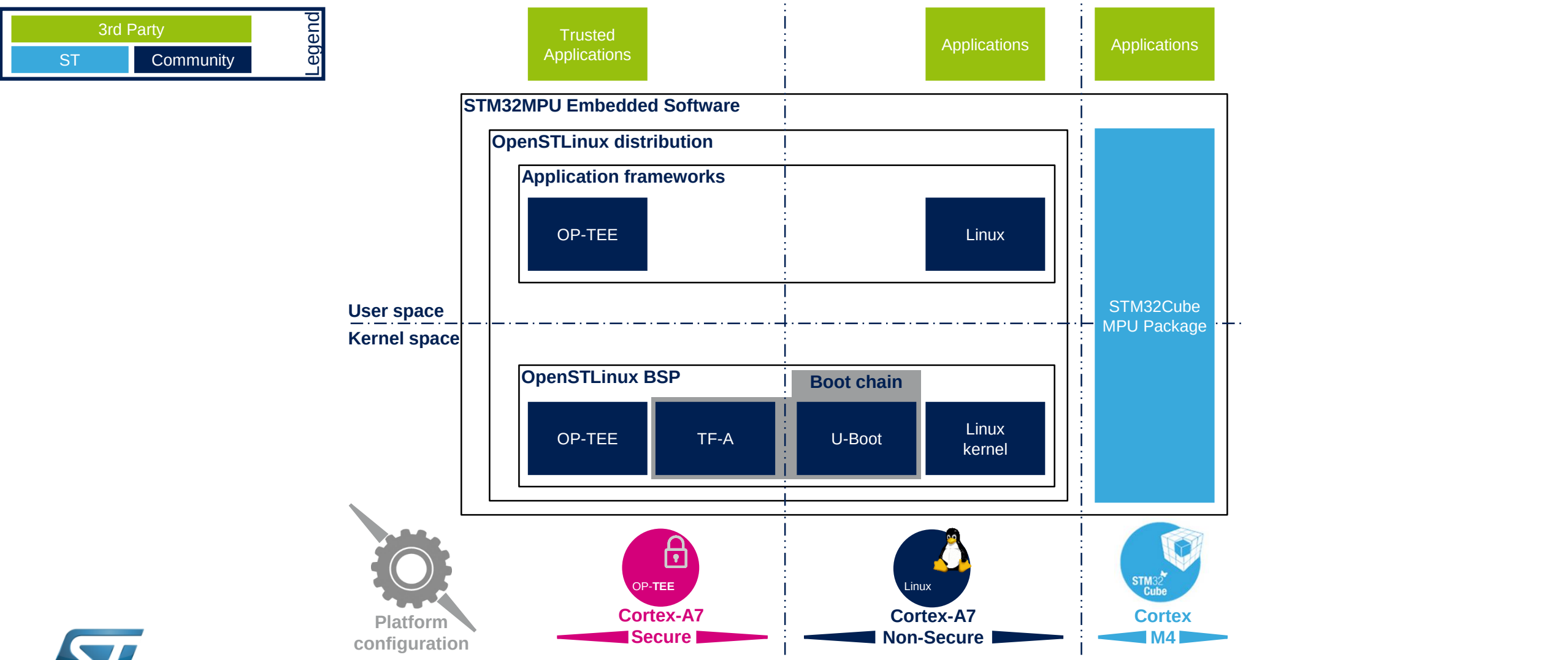
- Notice that each core may not see the same regions at the same address, as already explained on previous slide



- Each customer can of course tune this mapping (regions location and sizes) to fit with his product needs

life.augmented

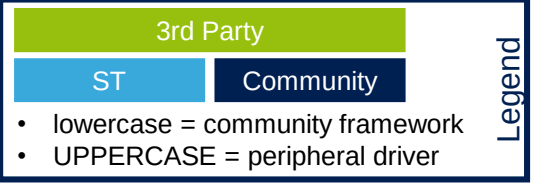
STM32MPU Embedded Software

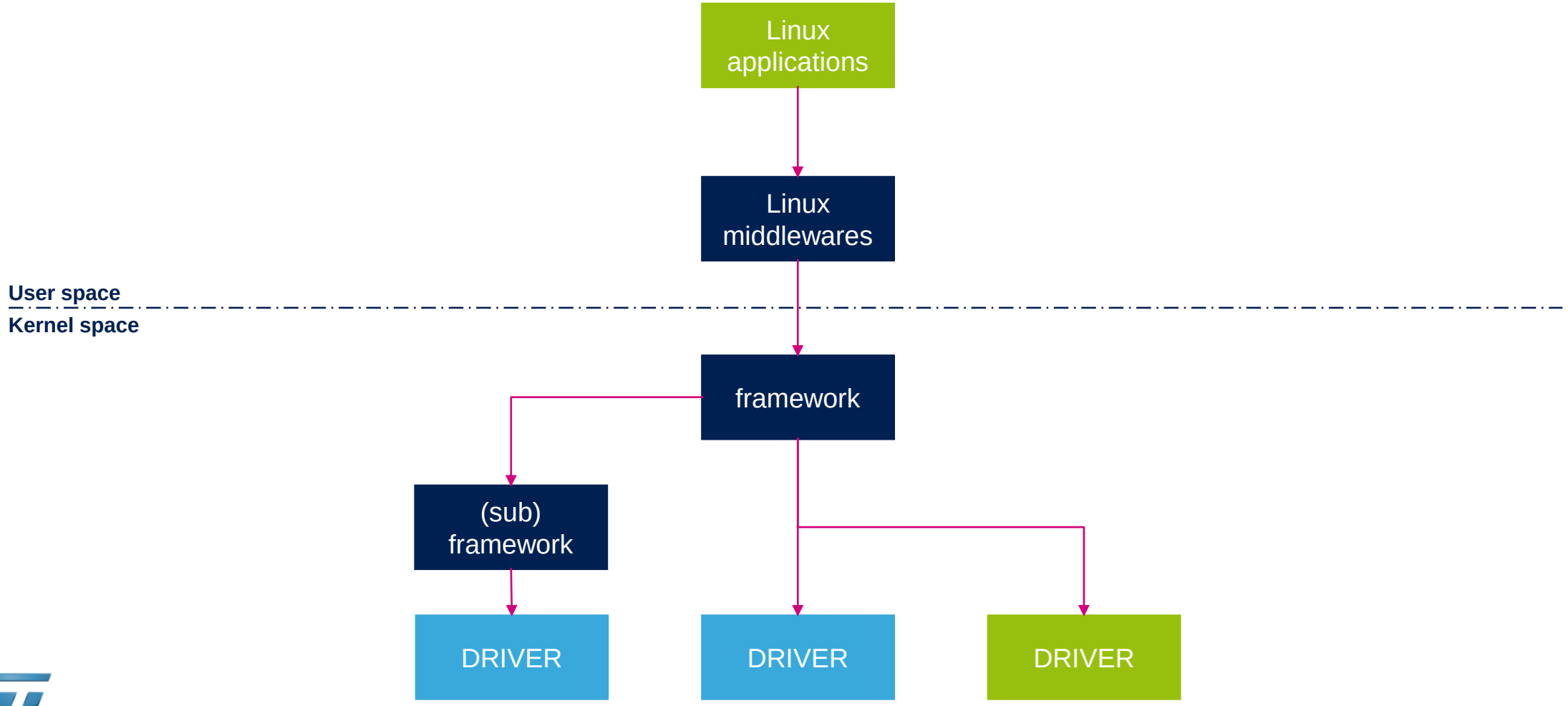


Source: ST Wiki article [STM32MPU Embedded Software architecture overview]

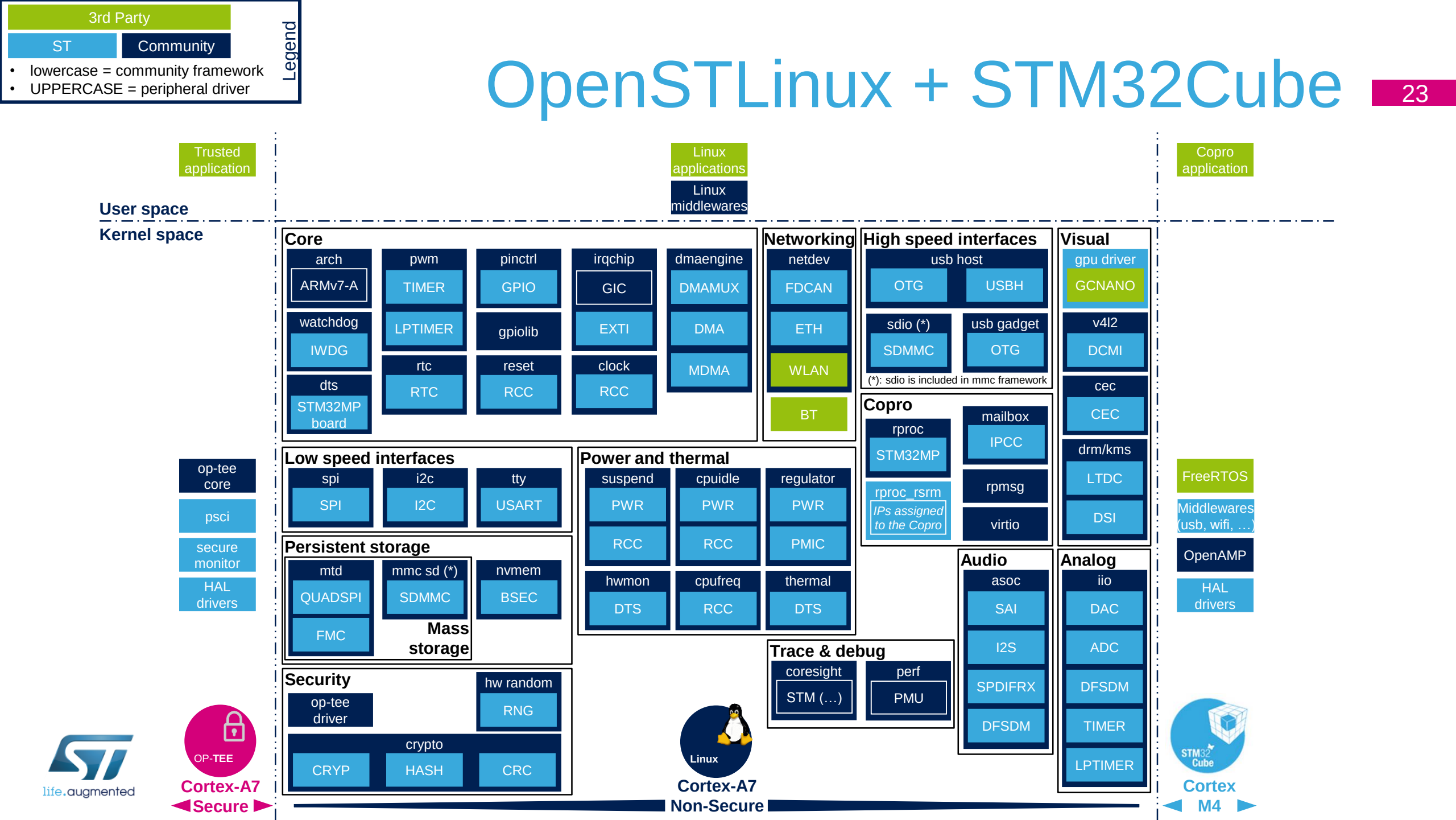
life.augmented

Linux framework & driver



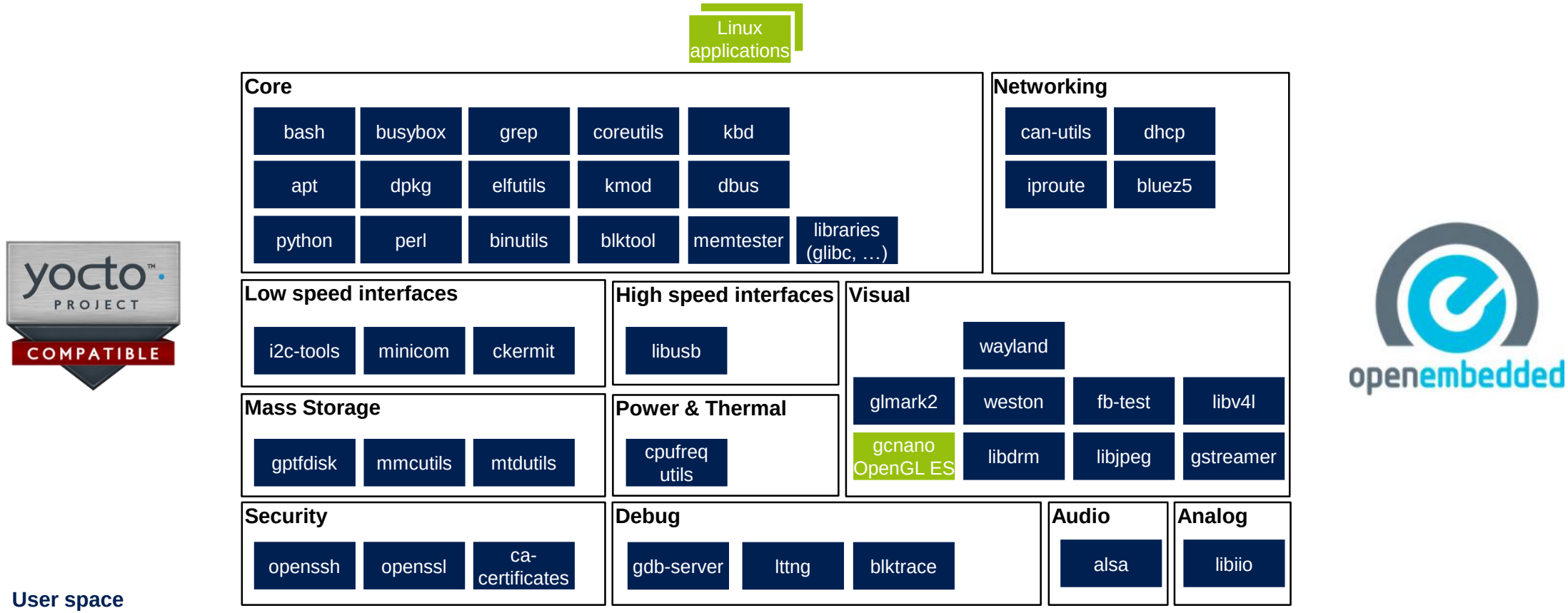


OpenSTLinux + STM32Cube



Open-Embedded User space

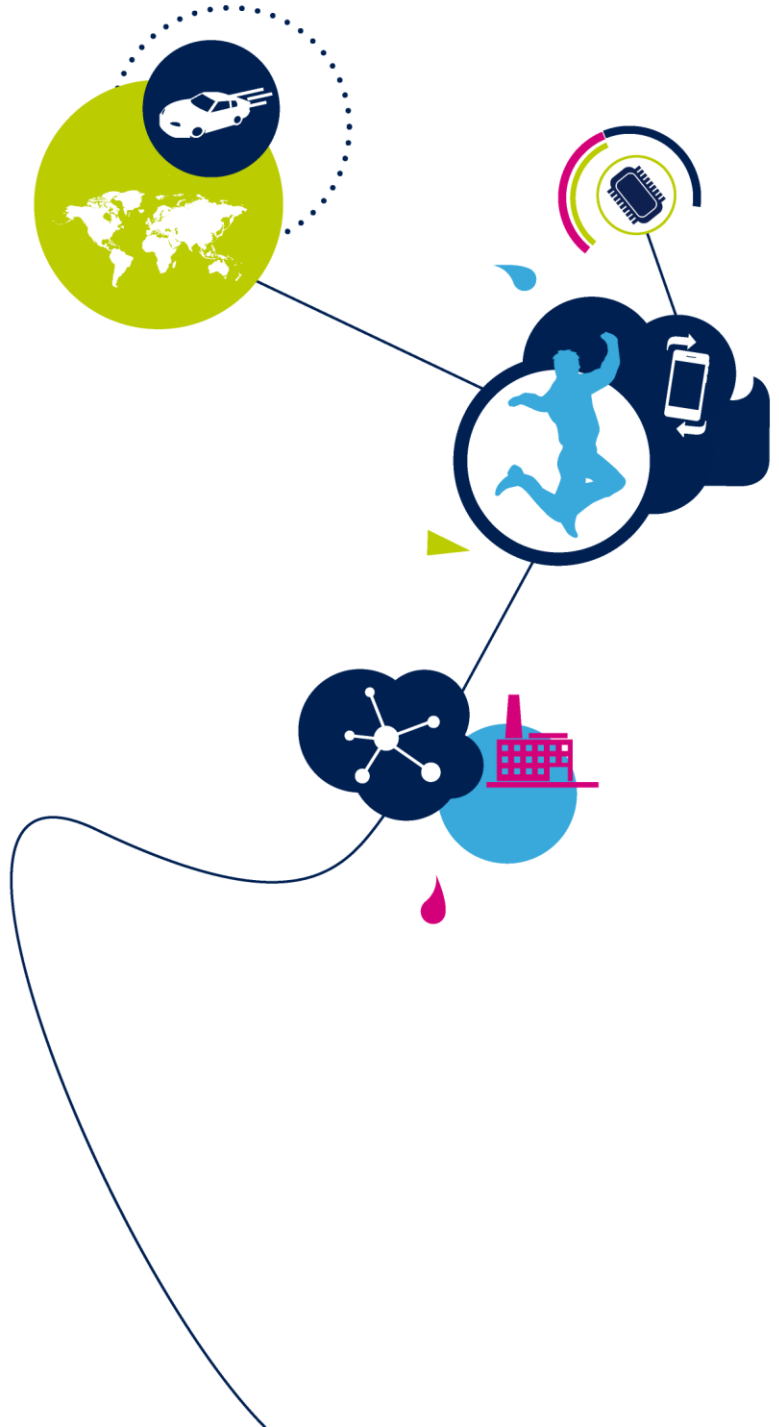
- The components list shown here is not exhaustive and can be tuned by the customer to fit with applications needs.



STM32MP1 Platform boot

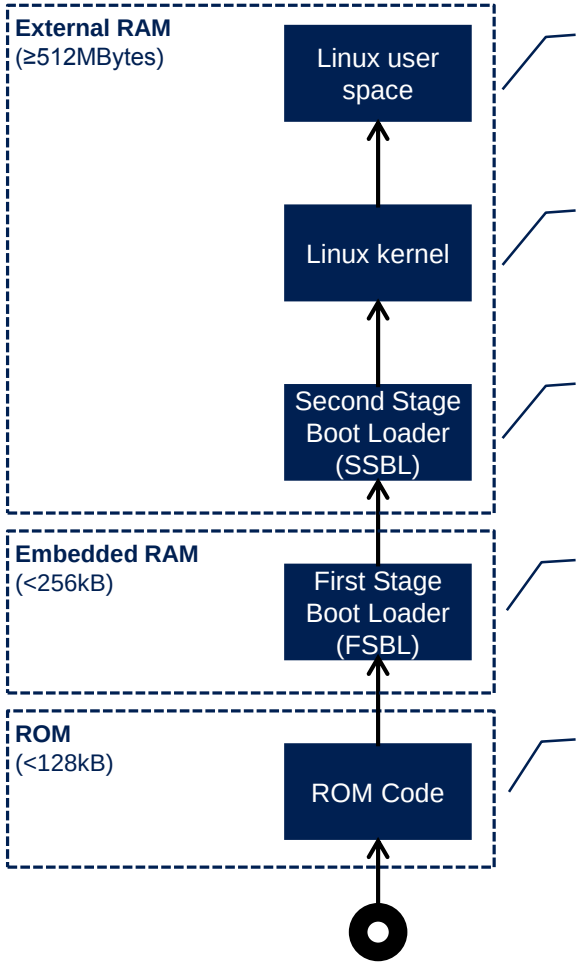
Platform boot

Revision 1.0



life.augmented

Standard Linux boot chain



- User space services and applications launch

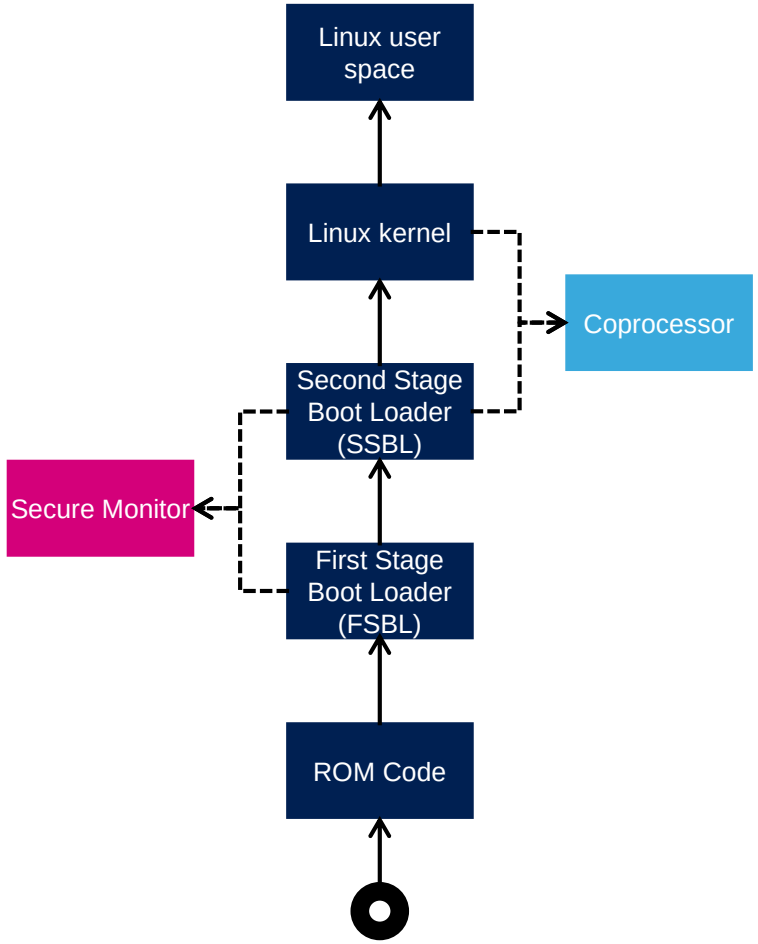
- Linux kernel initialisation (platform device drivers, etc.)
- Root file system (rootfs) mounting
- User space init process launch (/sbin/init)

- Boot file system (bootfs) loading from mass storage or Ethernet (TFTP)
- User feedback with boot loader splash screen
- Linux kernel (uImage) launch with its device tree blob (*.dtb)

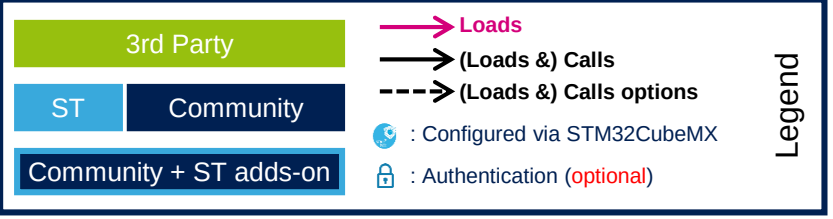
- Complete clock tree initialization
- External RAM (DDR, LpDDR) controller initialization
- SSBL loading from the boot device (mass storage or serial link)
- SSBL launch

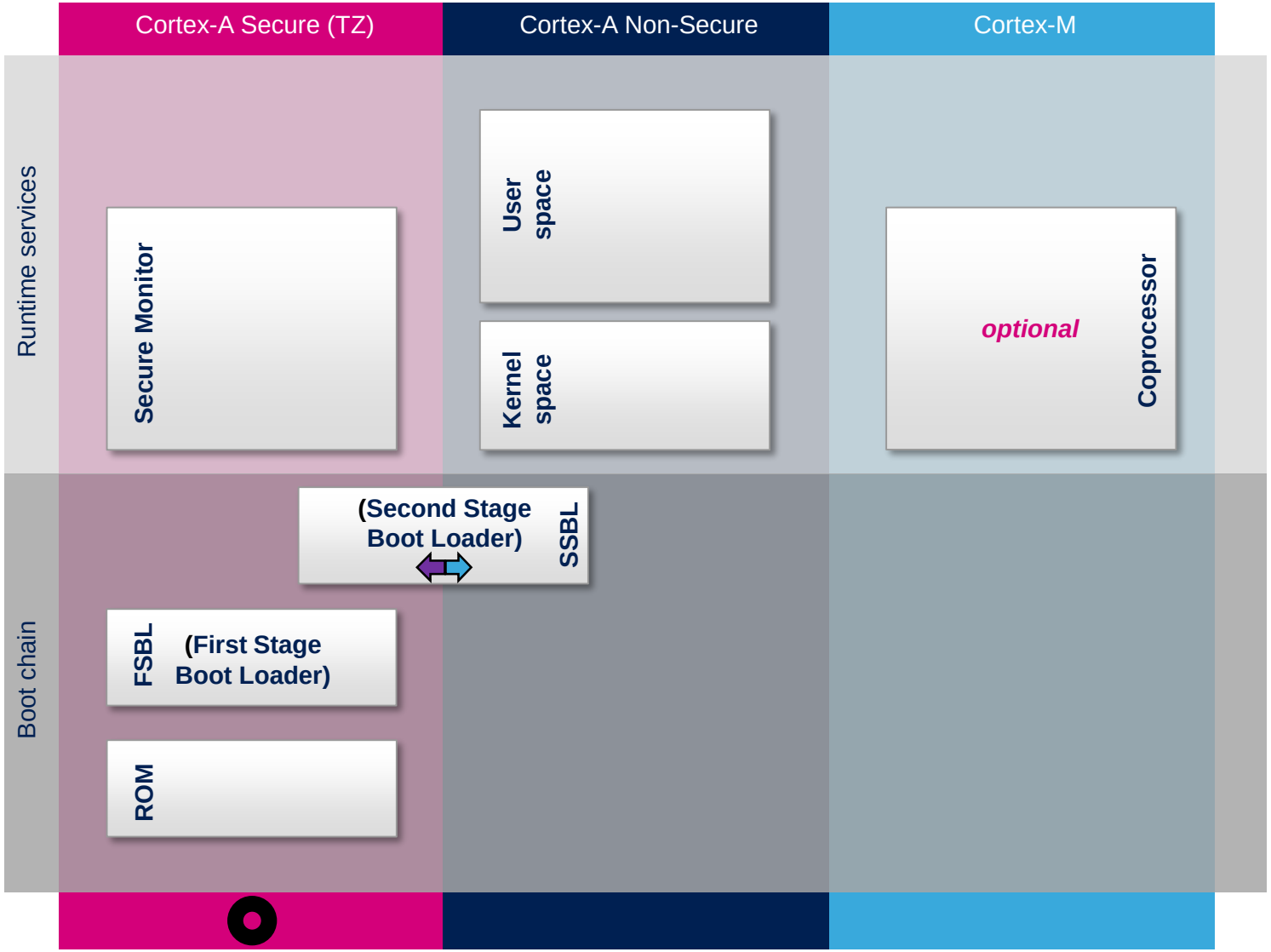
- Basic clock tree initialization
- FSBL loading from the boot device (mass storage or serial link)
- FSBL launch

STM32MP1 boot chain



STM32MP1 boot chains

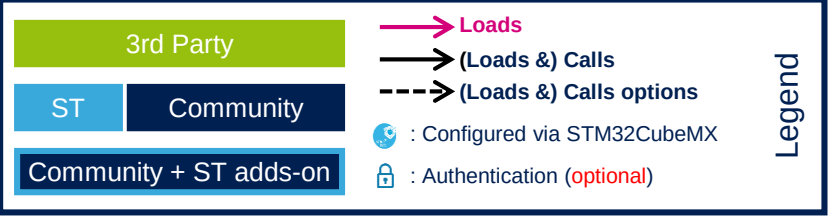


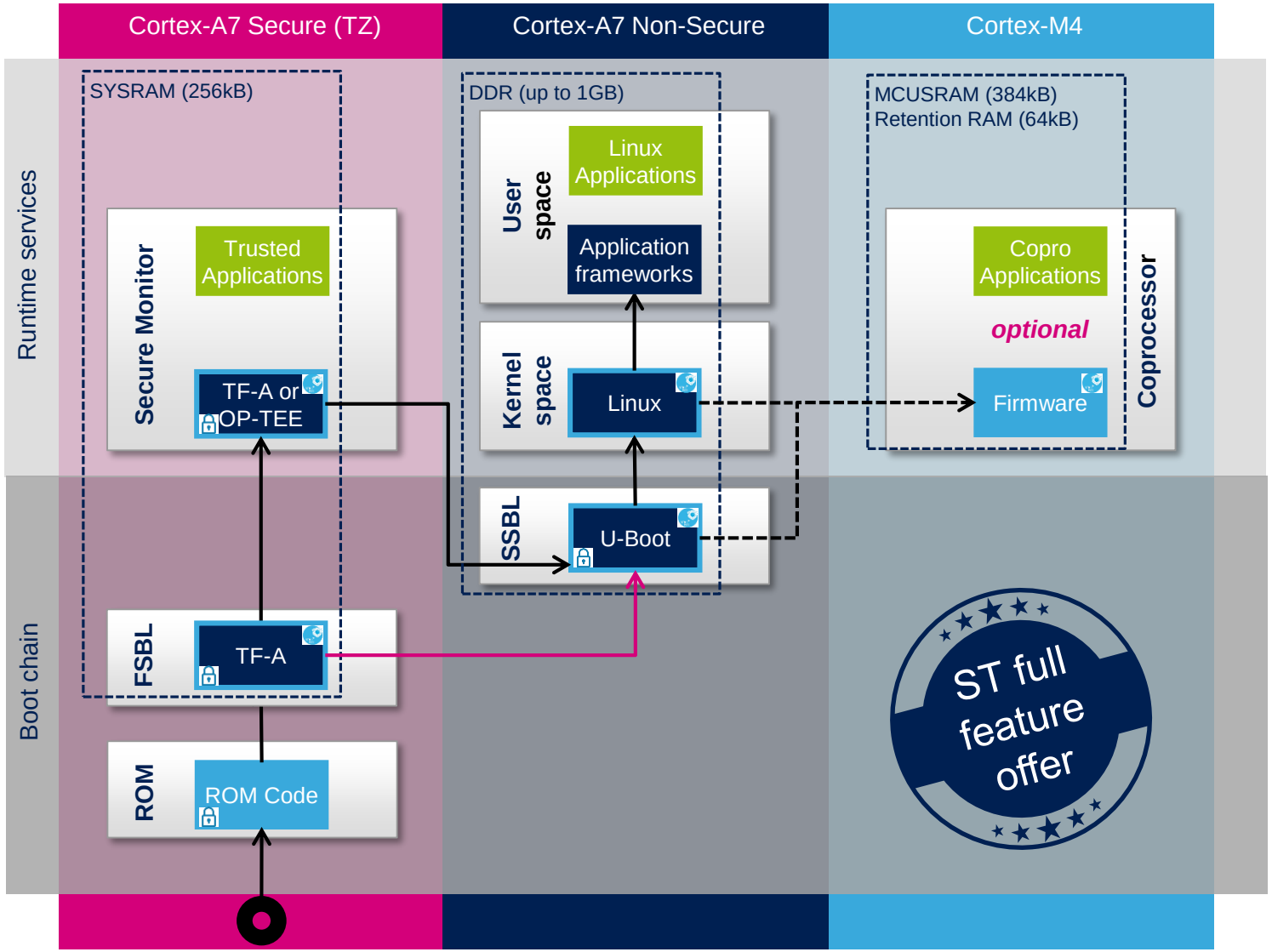


life.augmented

↔ : 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





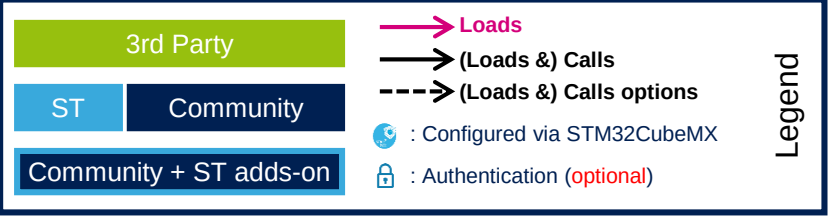
TF-A is:

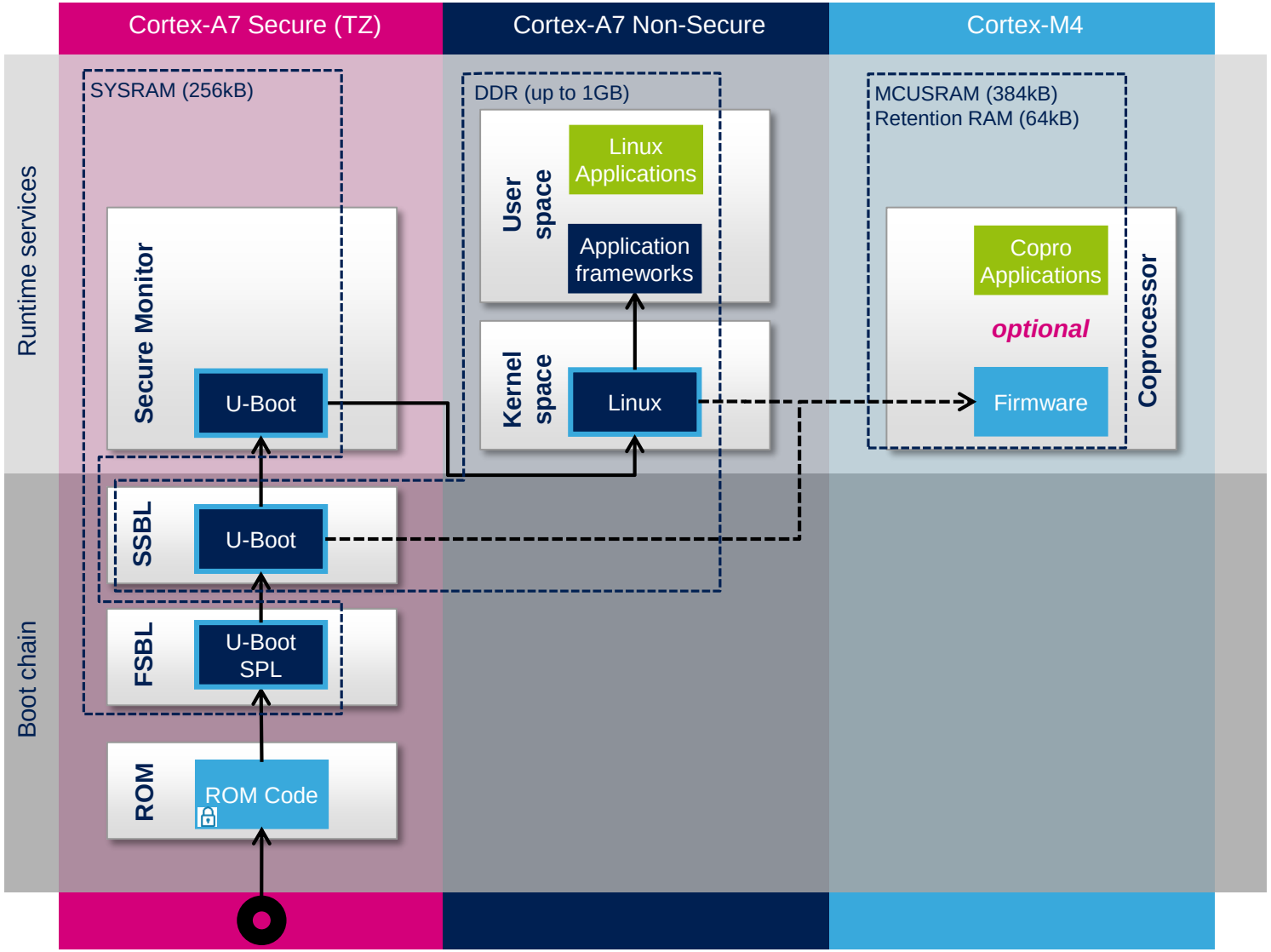
- BSD licence
- Trusted writing
- ARMv8 future proof

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	-	-

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

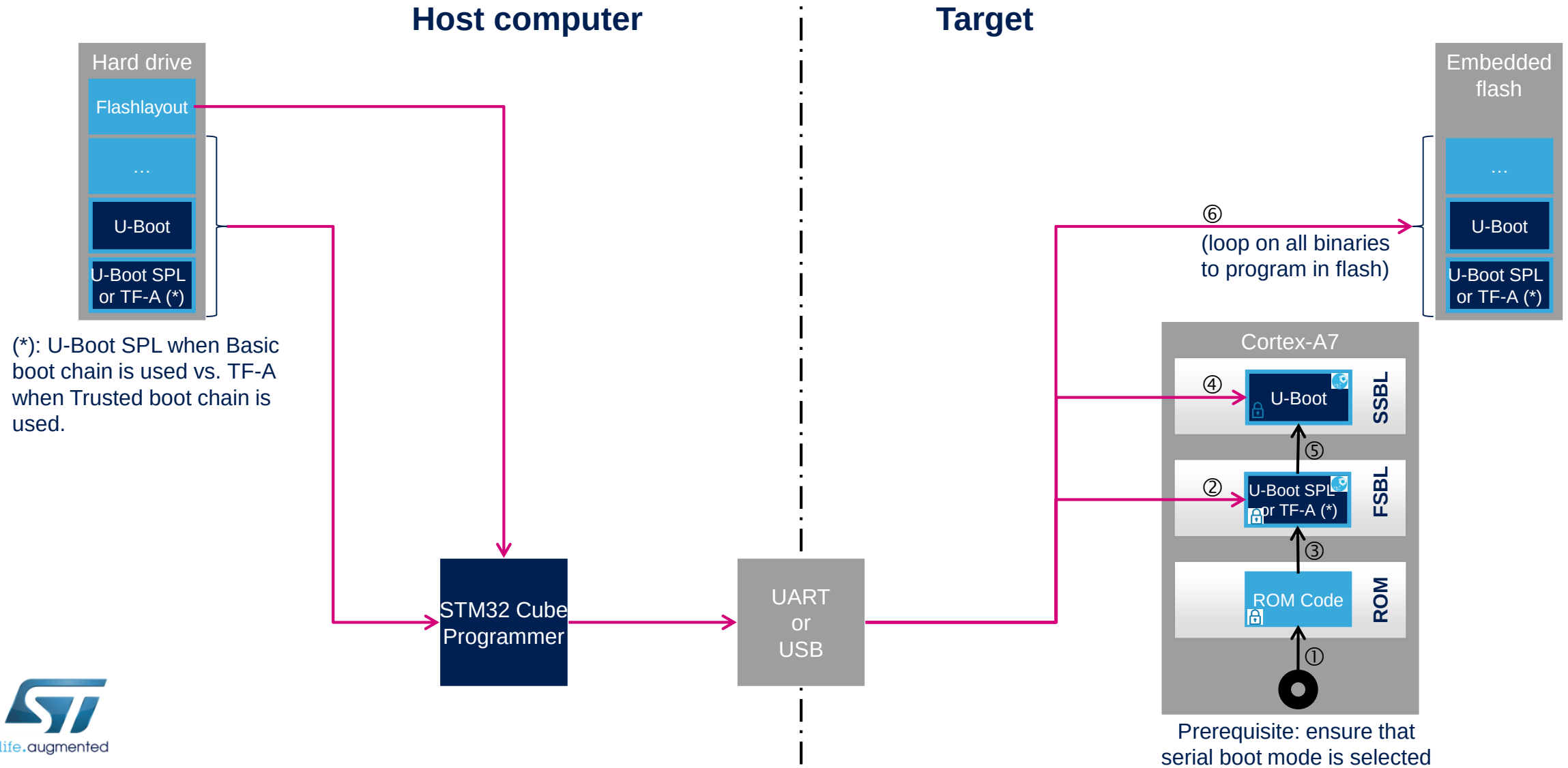
0	No secondary boot source is defined
1	FMC NAND

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

life.augmented

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

STM32CubeProgrammer for flash programming



life.augmented

STM32MP1 OpenSTLinux distribution

OpenSTLinux distribution overview

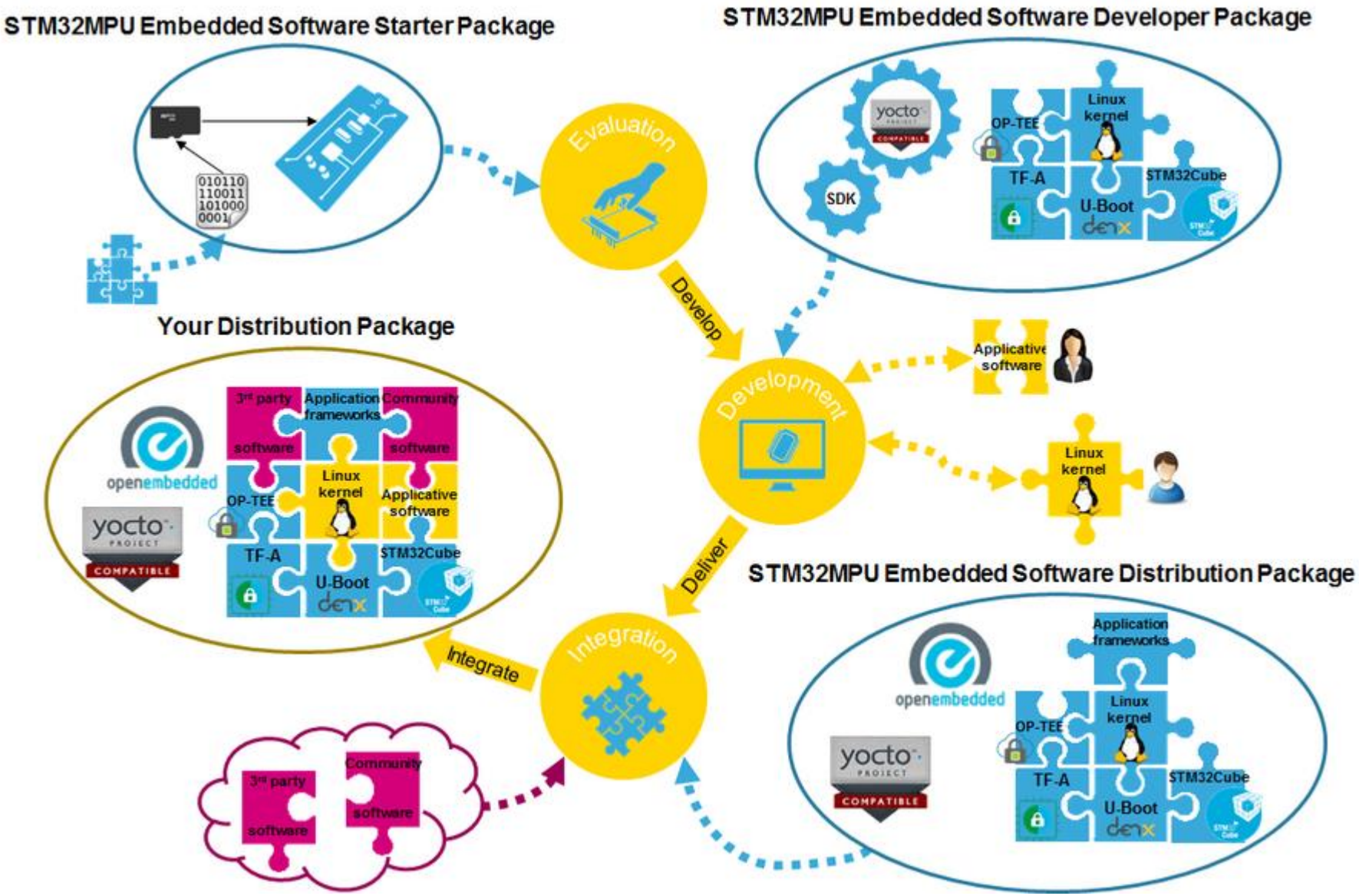
Revision 1.0

life.augmented

OpenSTLinux

- OpenSTLinux is a concept for STM32MPU embedded SW package
  - Concept = naming + associated pillars
- Pillars
  - Usage of a standard kernel interface (no proprietary interface)
  - Usage of Open Source software
  - Link to community (upstream)
  - Easy to use
- It supports OpenEmbedded build process
  - Yocto Compatible (target is to have a Board Support Package (BSP) hosted on Yocto server)

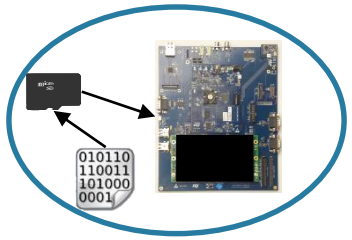
Available packages



life.augmented

Packages and use cases

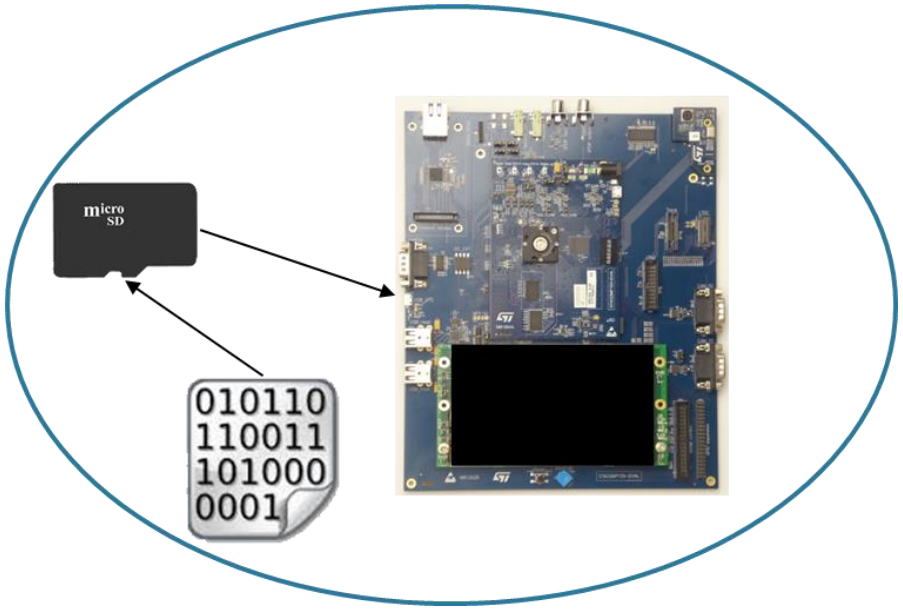
- Packages
  - Starter Package — Flashable images
  - Developer Package — Software Development Kit (SDK) and BSP tarballs
  - Distribution Package — Open Embedded distribution full source



- Rationale
  - Cover all the possible usages from our customers
    - Discover -> Prototype -> Start new hardware -> Productize software
  - Customer receives a board with an OpenSTLinux Starter package
    - Assess board capacities and performance
  - Customer wants to run existing applications = Starter package
  - Customer wants to develop their own application = Developer package
  - Customer wants to start their own hardware = prototype with Developer package then Distribution
  - Productization of software = Distribution package

Starter Package

- STM32MPU Embedded Software Starter Package
  - ST images stored with HW diversity flashlayout.zip = Yocto based images

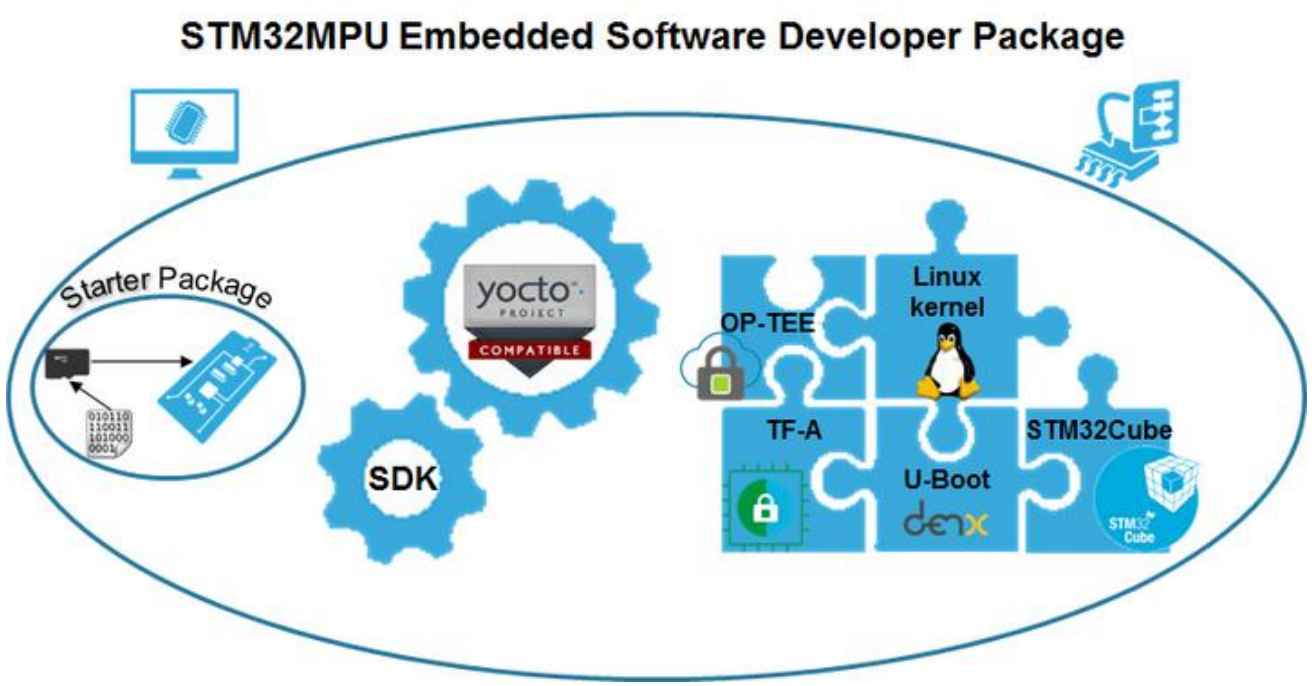


Focus on Starter Package

- Binaries ready to use by customer: Yocto/OpenEmbedded image (Weston)
  - Demonstrate the capacity of the platform
  - Available on microSD card or directly flashed onto the board
  - At this stage, Yocto starter package is only a prerequisite for developer kit
- Deliverables
  - Set of binaries with a specific flash layout
  - Script to generate an SD card raw image
  - Complete combinations (binaries x flash layouts) of microSD card ready to use images (aka stimg) are not provided by default; there are too many configurations

Developer Package

- Yocto based on st-image-weston =
  - SDK (Toolchain + Includes)
  - Sources (Community Tarball + ST Patches + ST configs)
    - Kernel
    - Boot (U-Boot, ATF)
  - STM32MP1-M4 Cube

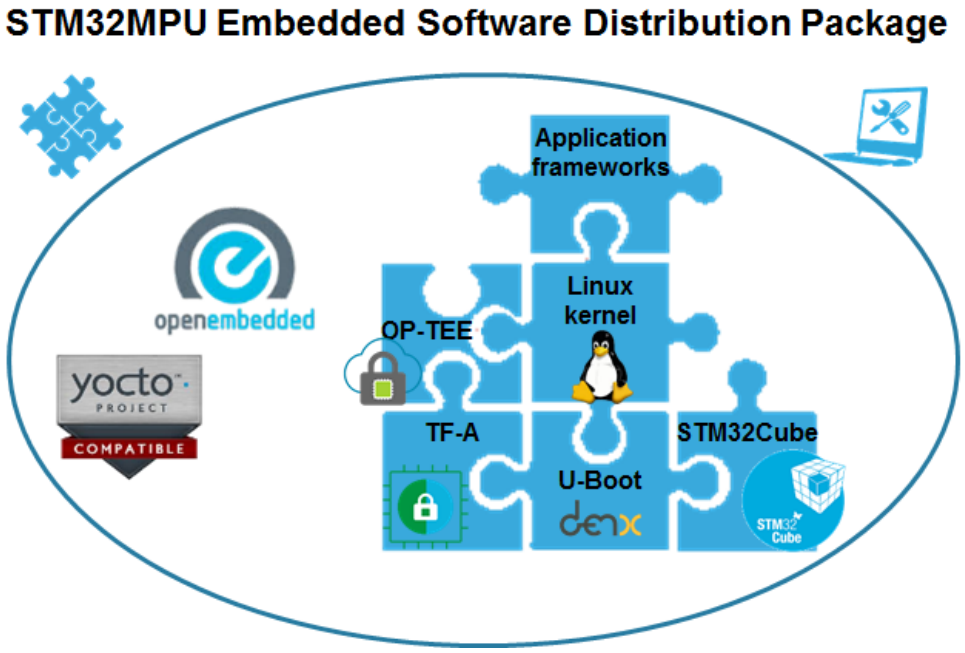


Focus on Developer Package

- Developer package = SDK (Software Development Kit) + Source code
  - Uses Starter package images
  - Only Yocto/OpenEmbedded SDK, based on Weston images, is provided
  - Source code provided
    - Kernel, U-Boot, ATF, Optee (optional), STM32Cube
  - Pre-compiled toolchain
- Release mode depends on the project stages
  - Alpha customer
    - Source code (tar ball from community) + patch
  - Mass market
    - Source code (tar ball from community) + patch
    - Git (ST github) = community content + all patches pending upstream

Distribution Package

- Yocto based - (ST layers + ST patches on Git community) =
  - Oe-manifest
  - meta-st-stm32mp
  - meta-st-openstlinux
  - STM32MP1-M4 Cube
  - meta-st-custo (to customise via appends per customer)



Focus on Distribution package

- Distribution package
  - Yocto/OpenEmbedded environment
    - OpenSTLinux BSP (kernel, ST drivers)
    - OpenSTLinux application framework (e.g. Weston, GStreamer)
  - OpenEmbedded recipes point to the developer git (SRC_URI)
  - Everything is to be compiled (toolchain, kernel, images)
- Kernel version
  - Alpha stage
    - Latest stable kernel + all patches pending upstream
  - Mass market
    - Alpha stage (delivered as example)
    - LTS kernel + all patches pending upstream (Default configuration)

OpenEmbedded

- Project initiated by the Linux Foundation in 2010 and is still managed by one of its fellows: Richard Purdie.
- Linux-based cross-compilation framework
- Open source (but can be used to build proprietary code)
- It is based on git for software configuration management

OpenEmbedded ecosystem

- People talk about Yocto, Poky or OpenEmbedded and this can be confusing:
  - OpenEmbedded:
    - Build Framework for embedded Linux
    - Maintained by the community
    - Source version of Poky
    - Setup mainly consolidated for ARM platforms
  - Yocto
    - A project that uses OpenEmbedded build system
  - Poky
    - Poky is a reference system of the Yocto Project - a collection of Yocto Project tools and metadata that serves as a set of working examples. Poky uses OpenEmbedded Core
    - Poky is maintained by Intel. Setup is mainly consolidated for Intel platforms
- Some projects works on a Yocto base, some others on a Poky base, but in the end everything is compatible.

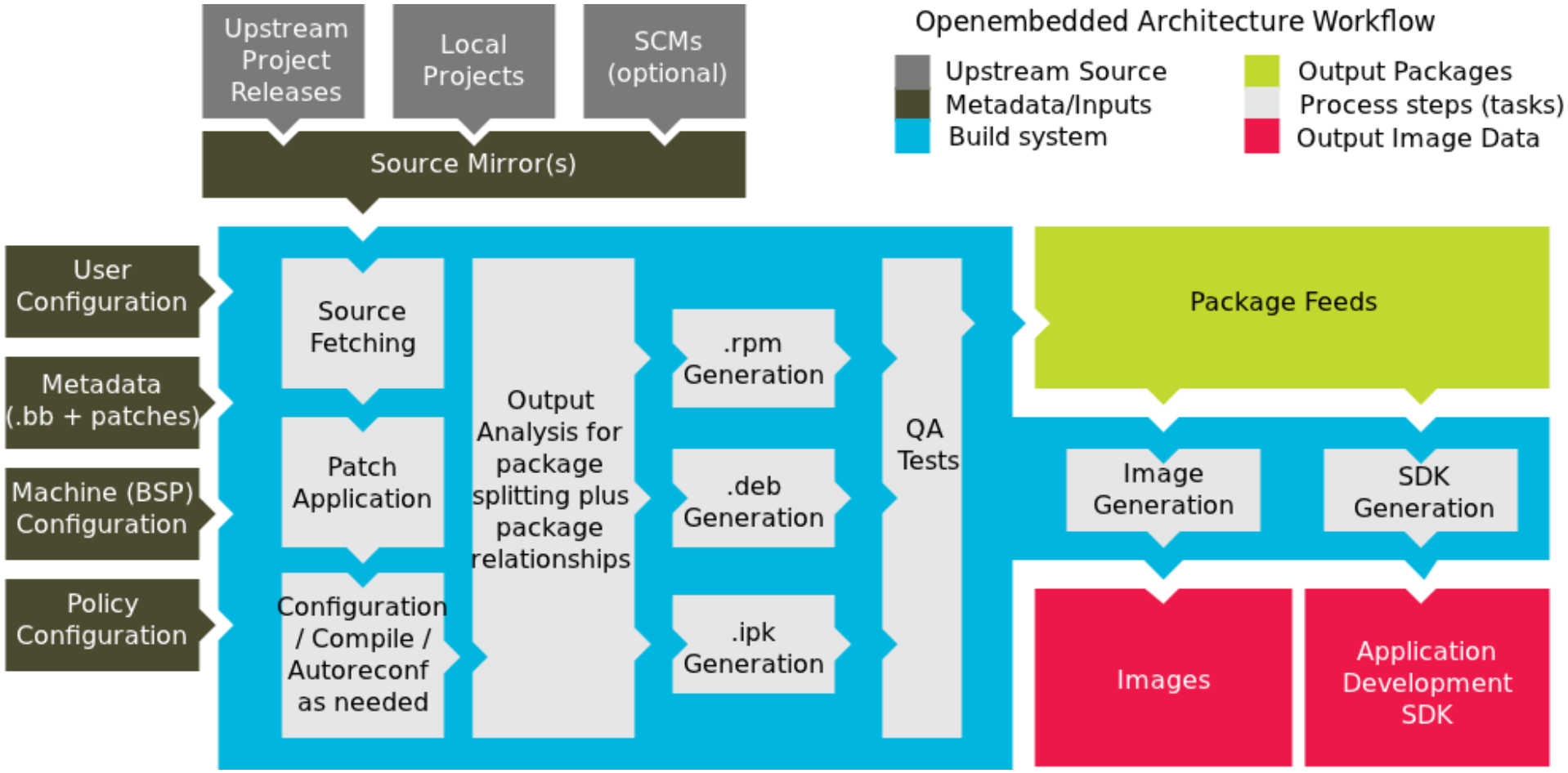
What OpenEmbedded does

- Source code download
- Patch application
- Cross compilation
- Package management

What OpenEmbedded generates

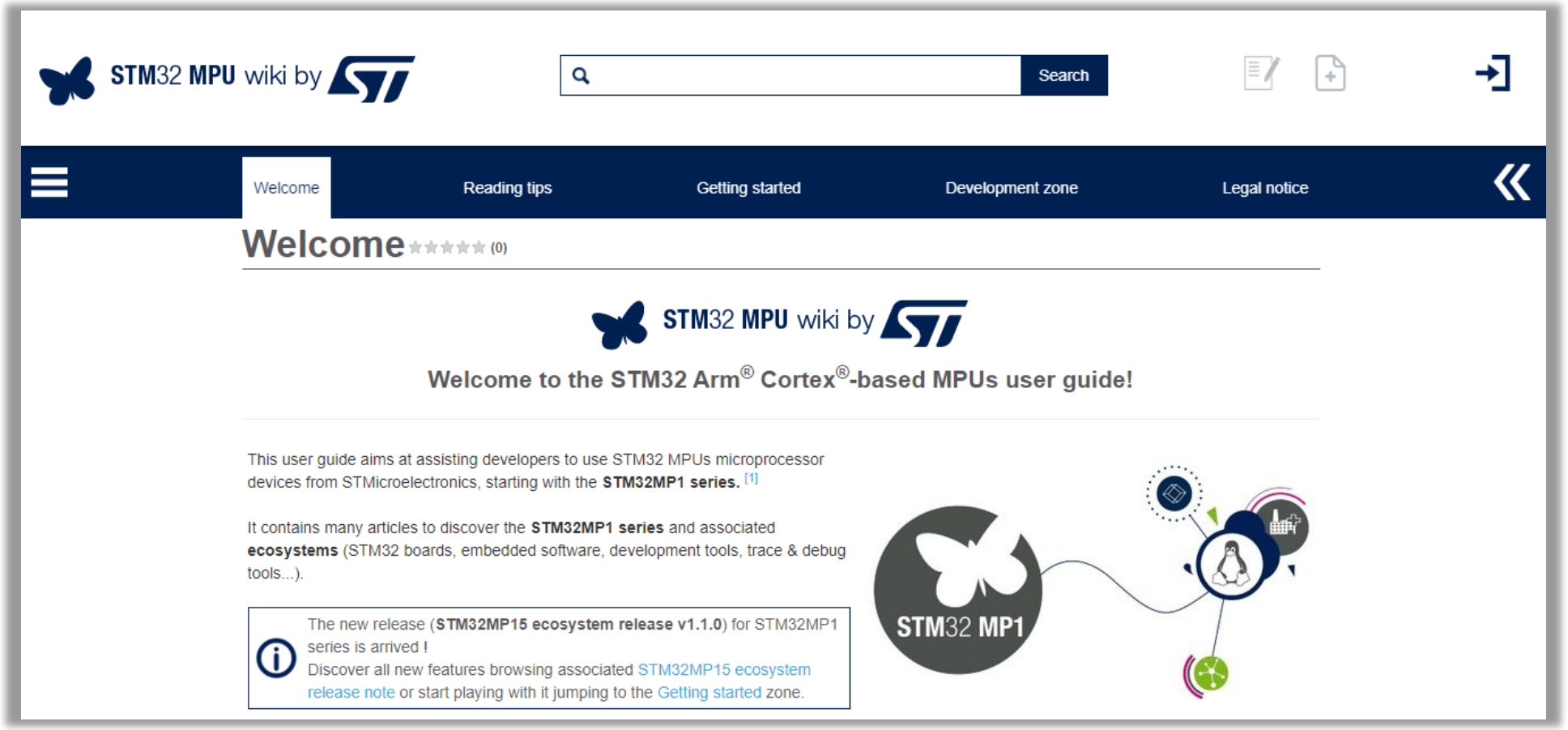
- Binary packages
- Linux-based system images
- Toolchains
- SDKs (Software Development Kits)

Compilation flow



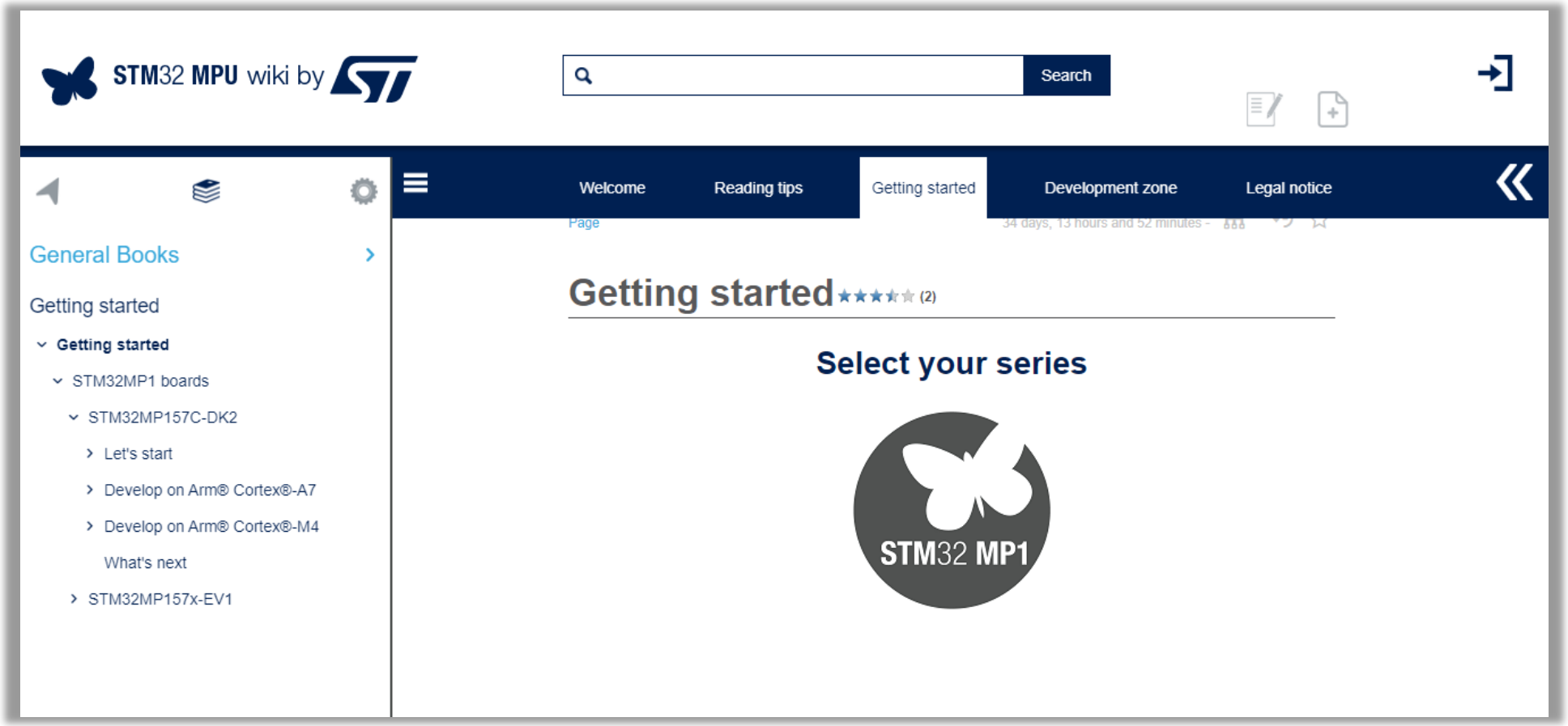
Useful links

- STM32MP1 Wiki: **https://wiki.st.com/stm32mpu/index.php/Main_Page**



Useful links

- STM32MP1 Wiki: **https://wiki.st.com/stm32mpu/index.php/Main_Page**



Thank you

STM32 MP1

STM32MP1 MPU series

www.st.com/stm32mp1

life.augmented