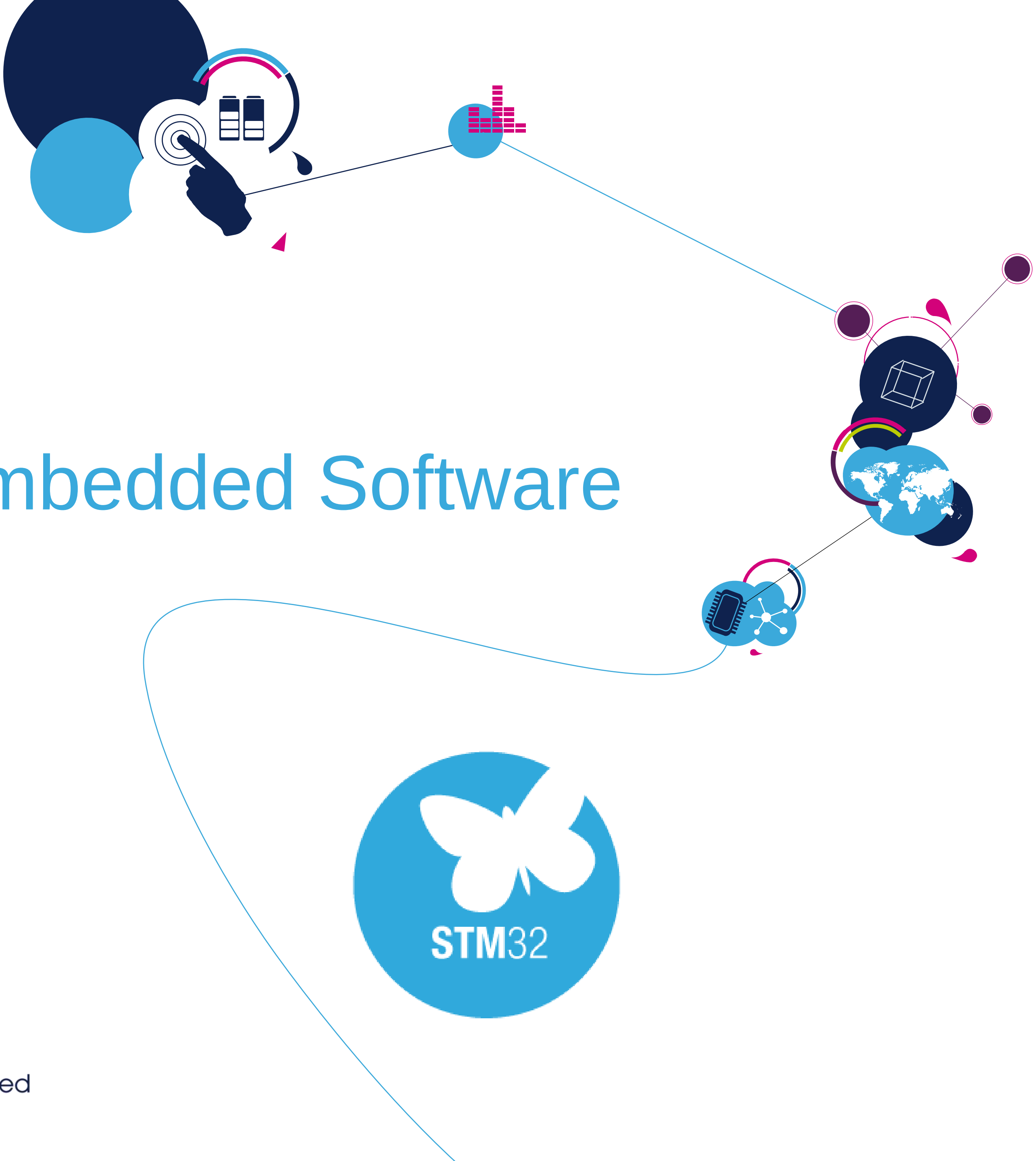
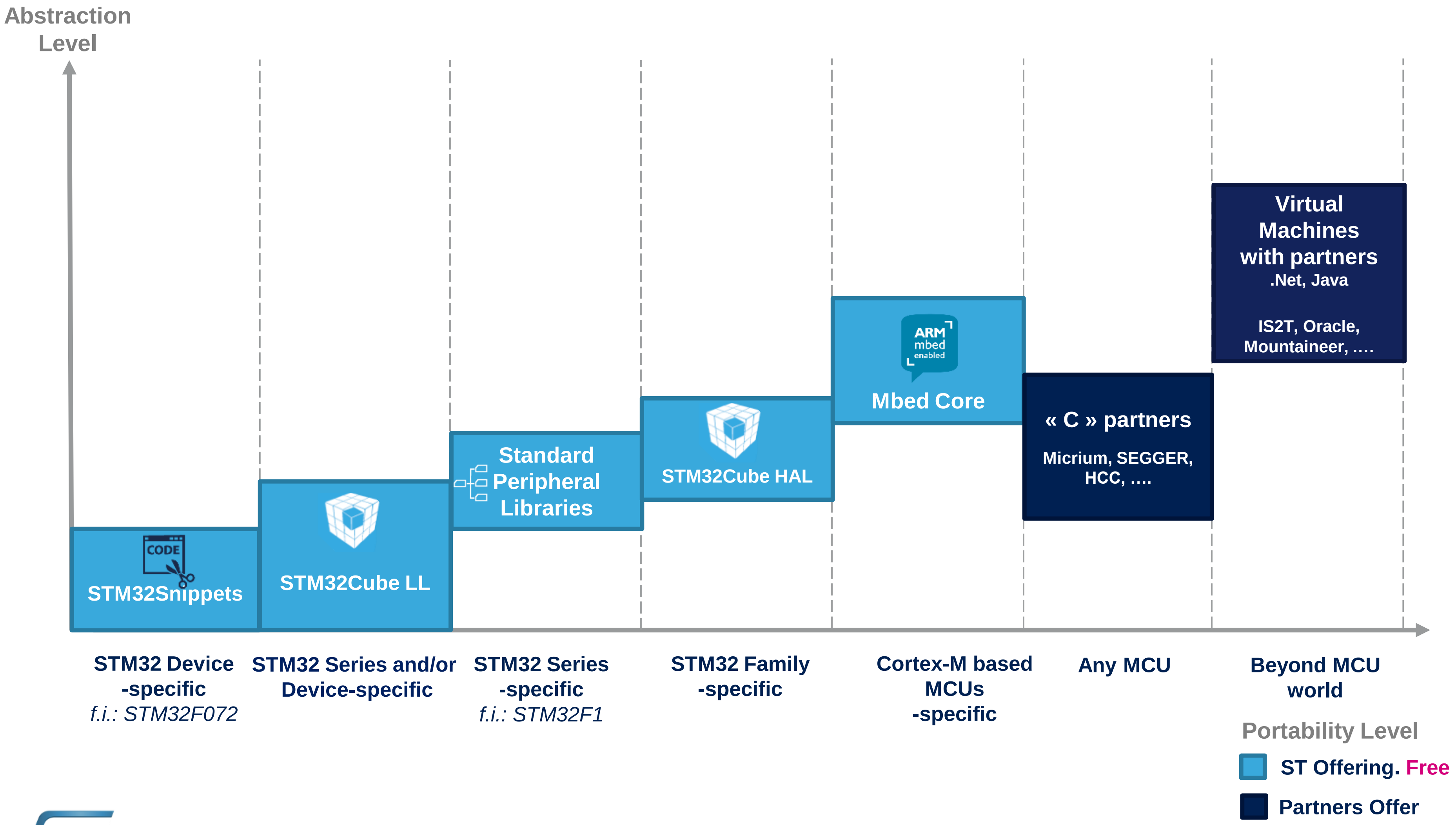
STM32 Embedded Software

Overview

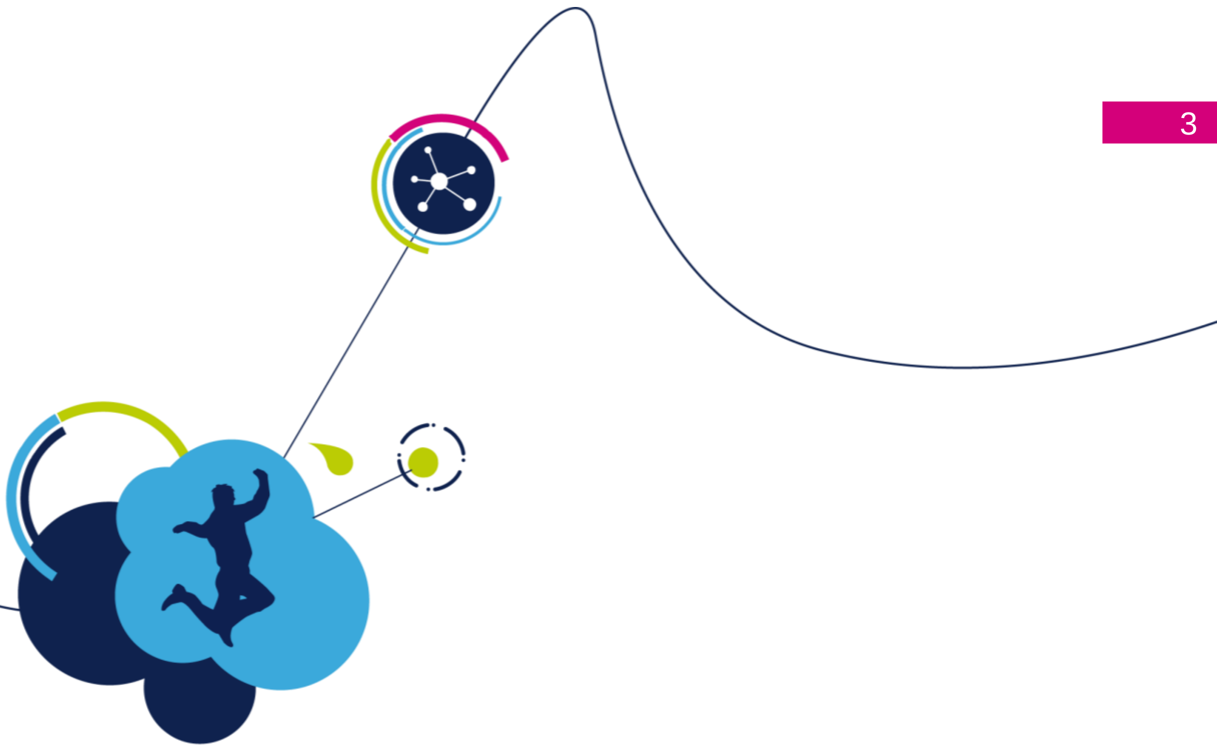


life.augmented

STM32 Embedded Software Offer



life.augmented



Focus on ST’s Offer (Free)

STM32Snippets

- What is it ?
  - A collection of code examples, directly based on STM32 peripheral registers, available in documentation and as software bundles
- Target Audience
  - low level embedded system developers, typically coming from an 8 bit background, used to assembly or C with little abstraction
- Features:
  - Highly Optimized
  - Register Level Access
  - Small code expressions
  - Closely follows the reference manual
  - Debugging close to register level
- Limitations:
  - Specific to STM32 devices, not portable directly between series
  - Not matching complex peripherals such as USB
  - Lack of abstraction means developers must understand peripheral operation at register level
  - Available (today) on STM32 L0 and F0 series

Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
	+++			+

life.augmented

Standard Peripheral Libraries (SPL)

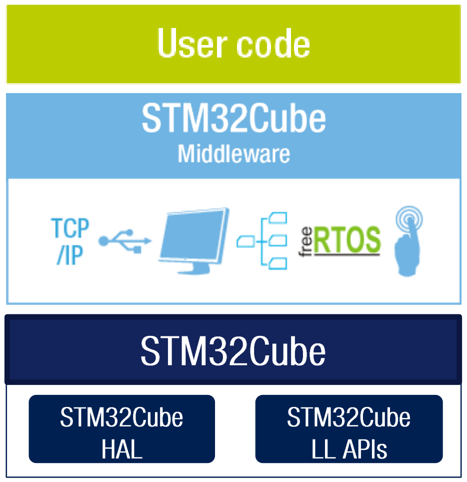
- What is it ?
  - Collection of C Libraries covering STM32 peripherals
- Target Audience
  - Embedded systems developers with procedural C background. All existing STM32 customer base prior to the STM32Cube launch, willing to keep same supporting technology for future projects, and same STM32 series.
- Features:
  - Average optimization, fitting lots of situations
  - No need for direct register manipulation
  - 100% coverage of all peripherals
  - Easier debugging of procedural code
  - Extensions for complex middleware such as USB/TCP-IP/Graphics/Touch Sense
- Limitations:
  - Specific to certain STM32 series.
  - No common HAL API prevents application portability between series
  - Middleware libraries may not be unified for each series
  - Doesn’t support forward STM32 series starting with STM32 L0, L4 and F7

Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
++	++	+	++	+++

STM32Cube - Embedded software

Introduction

- What is it ?
  - Full featured packages with drivers, USB, TCP/IP, Graphics, File system and RTOS
  - Set of common application programming interfaces, ensuring high portability inside whole STM32 family
  - Set of APIs directly based on STM32 peripheral registers
  - Set of initialization APIs functionally similar to the SPL block peripheral initialization functions

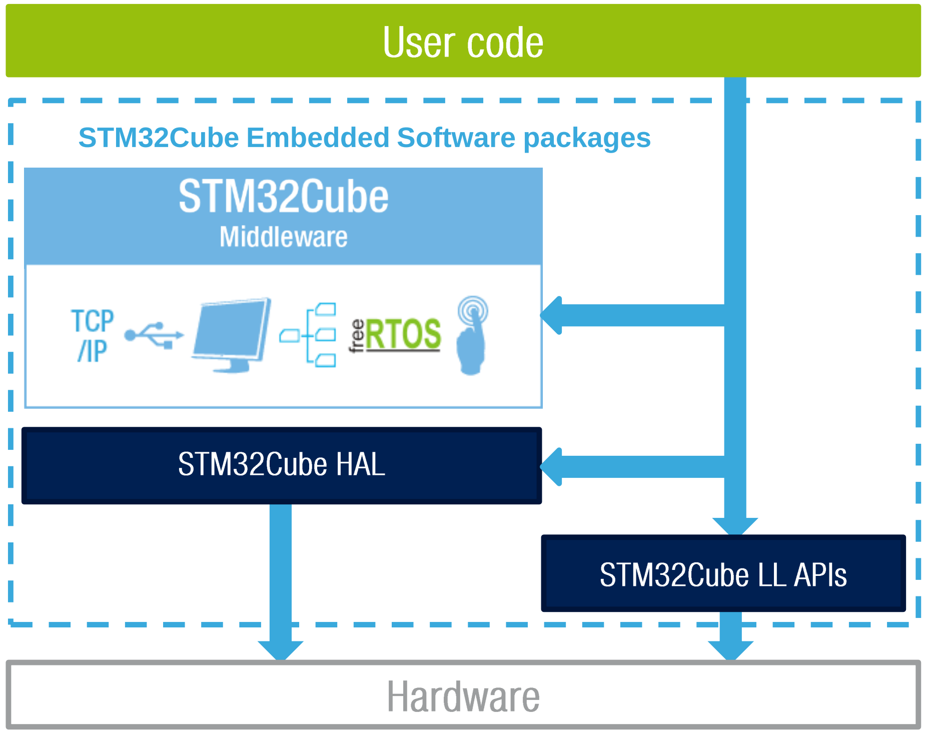


- Target Audience
  - Hardware Abstraction Layer (HAL) APIs: embedded system developers with a strong structured background. New customers looking for a fast way to evaluate STM32 and easy portability
  - Low-Layer (LL) APIs: low level embedded system developers, typically coming from an 8-bit background, used to assembly or C with little abstraction. Stronger focus on customers migrating from the SPL environment.

STM32Cube - Embedded software

Architecture overview

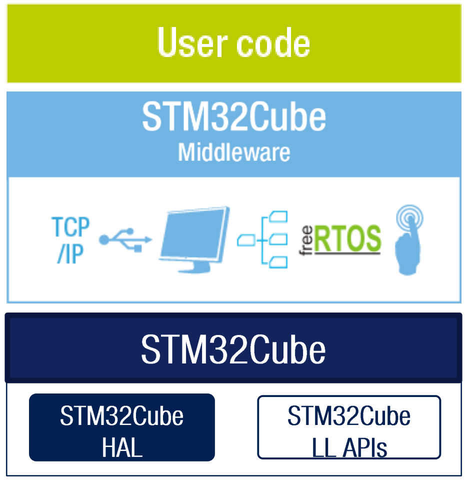
- Three entry points for the user application:
  - Middleware stacks
  - HAL APIs
  - LL APIs
- Possible concurrent usage of HAL and LL
  - Limitation: LL cannot be used with HAL for the same peripheral instance. Impossible to run concurrent processes on the same IP using both APIs, but sequential use is allowed
  - Example of hybrid model:
    - Simpler static peripheral initialization with HAL
    - Optimized runtime peripheral handling with LL calls



STM32Cube - Embedded software

HAL APIs

- Features:
  - High level and functional abstraction
  - Easy port from one series to another
  - 100% coverage of all peripherals
  - Integrates complex middleware such as USB/TCP-IP/Graphics/Touch Sense/RTOS
  - Can work with STM32CubeMX tool on the PC to generate initialization code
- Limitations:
  - May be challenging to low level C programmers in the embedded space.
  - Higher portability creates bigger software footprints or more time spent executing adaptation code

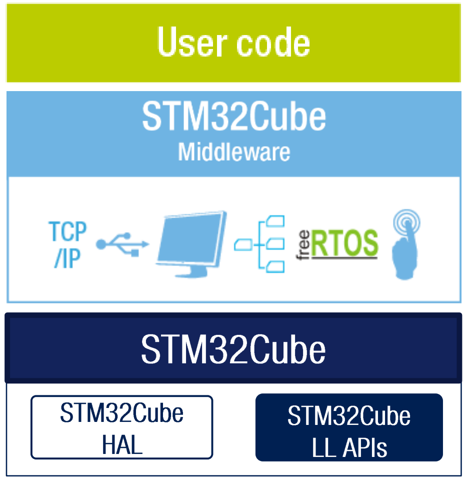


Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
+++	+	++	+++	+++

STM32Cube - Embedded software

Low-Layer APIs

- Features:
  - Highly Optimized
  - Register Level Access
  - Small code expressions
  - Closely follows the reference manual
  - Debugging close to register level
  - Peripheral block initialization APIs
    - Initialization, de-initialization and default initialization routines
    - SPL-Like functionally speaking
    - More optimized than SPL, fitting lots of situations
    - No need for direct register manipulation
    - Easier debugging of procedural code



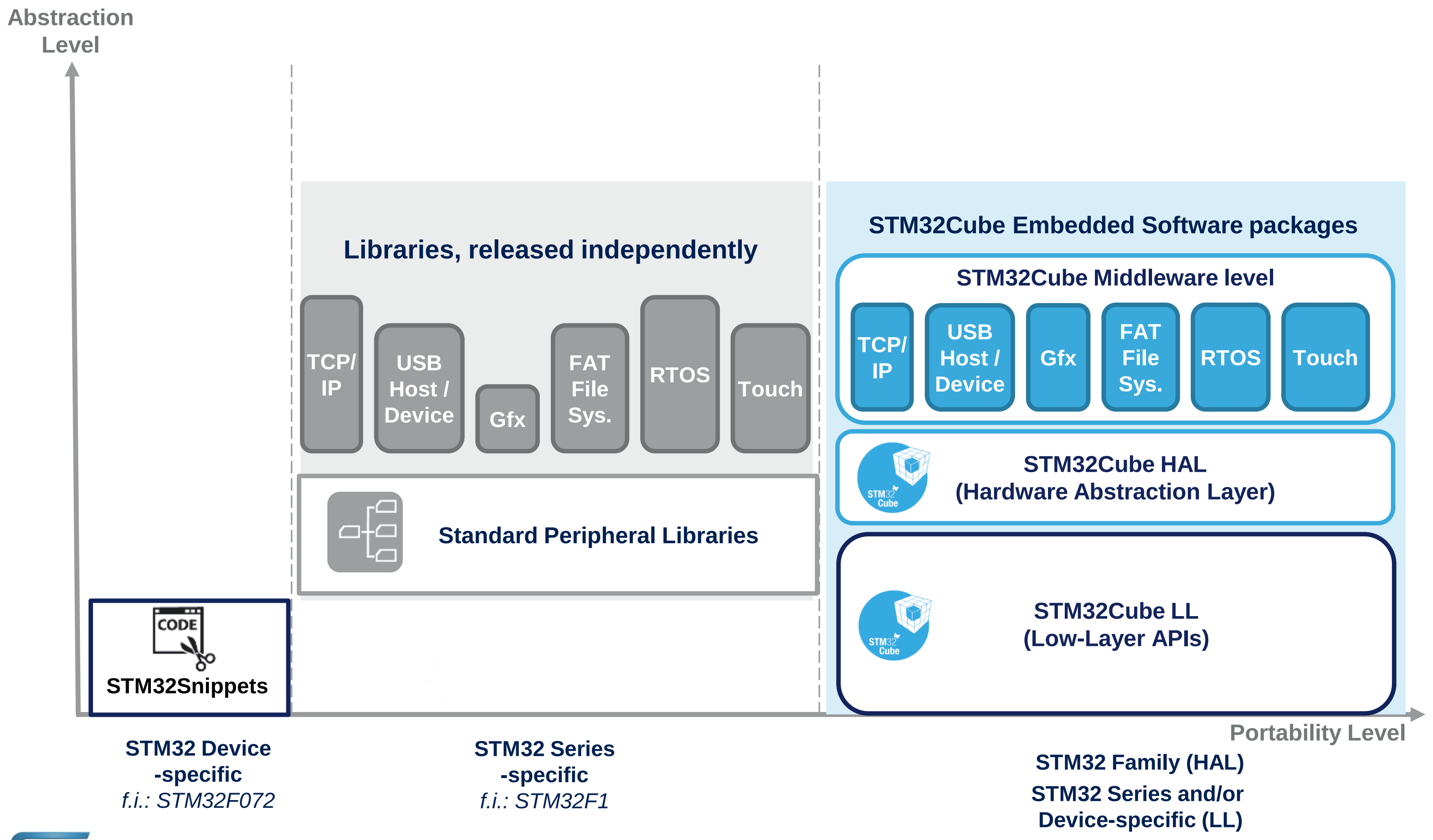
- Limitations:
  - Specific to STM32 devices, not portable directly between series
  - Not matching complex peripherals such as USB
  - Lack of abstraction for runtime means developers must understand peripheral operation at register level
  - Available on STM32L4, L0 and F0 series
  - Peripheral block initialization APIs have the same limitations as the SPLs (except availability considerations)

Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
+	+++	+	++	++

ST Embedded software offer – Comparison

Offer		Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
STM32Snippets			+++			+
Standard Peripheral Library		++	++	+	++	+++
STM32Cube	HAL APIs	+++	+	++	+++	+++
	LL APIs	+	+++	+	++	++

ST Embedded software offer – Positioning



Availability

Offer	Available for STM32								
	STM32 F0	STM32 F1	STM32 F3	STM32 F2	STM32 F4	STM32 F7	STM32 L0	STM32 L1	STM32 L4
STM32Snippets	Now	N.A.	N.A.	N.A.	N.A.	N.A.	Now	N.A.	N.A.
Standard Peripheral Library	Now	Now	Now	Now	Now	N.A.	N.A.	Now	N.A.
STM32Cube HAL	Now	Now	Now	Now	Now	Now	Now	Now	Now
STM32Cube LL NEW	Now	Q1 2017	Now	Q1 2017	Q1 2017	Q4 2016	Now	Now	Now

life.augmented

What solution to choose ? An FAQ

1. I want to use a small footprint MCU, what should I use?

   Abstraction has a cost. Therefore, if you need to take benefit from every single bit of memory, STM32Snippets or STM32Cube LL will be the best choice.

2. I come from 8-bit MCU world, what should I use?

   If you prefer direct register manipulation then the STM32Snippets or STM32Cube LL would be a good starting point. However, if you prefer structure 'C' level programming, then we recommend using the STM32Cube HAL or SPL.

3. I today use SPL on STM32F103. Should I switch to STM32Cube?

   If you intend to use only MCUs that are part of the same series in the future (in this case STM32 F1 series), then you should remain using SPL.

   If you plan to use different STM32 series in the future then we recommend considering STM32Cube as this will make it much easier to move between series.

4. I need a mix of portability and optimization. What can I do?

   You can use STM32Cube HALs and replace some of the calls with your specific optimized implementations, thus keeping maximum portability and isolating areas that are not portable, but optimized.

   HALs and LL being partially usable concurrently (no possible concurrent runtime HAL and LL processes for the same peripheral), it is also possible to use a hybrid HAL and LL implementation to get the same advantages as mentioned above.

life.augmented

Migrating between offers

		To		
From		**STM32Snippets**	**SPL**	**STM32Cube**
STM32Snippets		Easy within same STM32 series Ex: Between STM32F072 and STM32F030	No simple migration path. Application must be rewritten	HAL API: No simple migration path. Application must be rewritten
		Almost not possible between different series Ex: Between STM32F072 and STM32L053		Low-Layer API: Easy within same STM32 series Ex: Between STM32F072 and STM32F030
Standard Peripheral Library (SPL)		Some (but not all) SPL functions can be replaced with Snippets	Easy within same STM32 series Ex: Between STM32F401 and STM32F429	HAL API: No simple migration path. Application must be rewritten
			Difficult between different STM32 series Ex: Between STM32F100 and STM32F407	LL API: functionally equivalent functions vs SPL peripheral initialization functions
STM32Cube embedded software package	HAL API	Some (but not all) HAL functions can be replaced with Snippets	No simple migration path. Application must be rewritten	HAL: Yes, across all STM32 families
	LL API	LL calls equivalent to snippets when addressing the exact same peripheral	SPL block peripheral initialization functions have functionally equivalent functions in LLs	LL API: Difficult between different STM32 series Ex: Between STM32F407 and STM32L476

life.augmented

Thank you

/STM32 @ST_World st.com/e2e

www.st.com/stm32embeddedsoftware

life.augmented