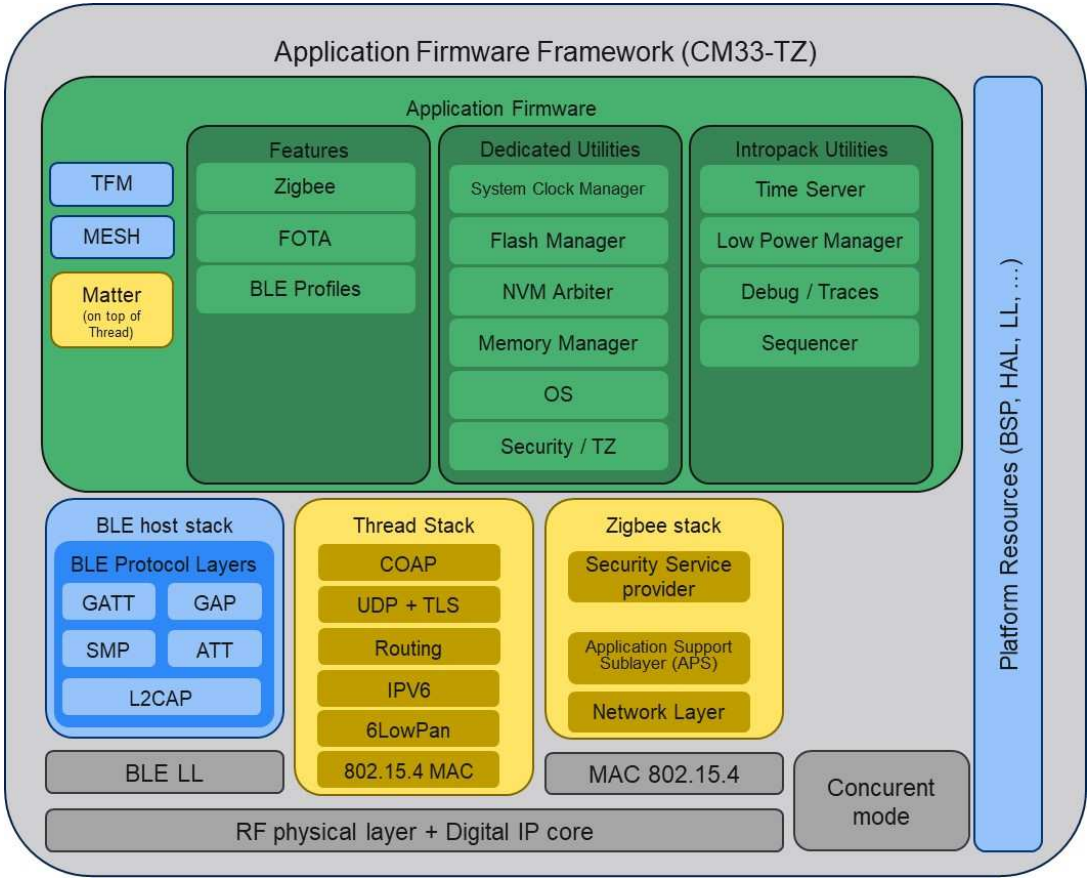
STM32WBA

SW architecture

STM32WBA overall SW Architecture



Multi stack support

BLE 5.3 full features
Zigbee*
Thread*
Matter *

Utilities to ease and secure your design

Low Power Manager
Sequencer

Platform resources to address your HW

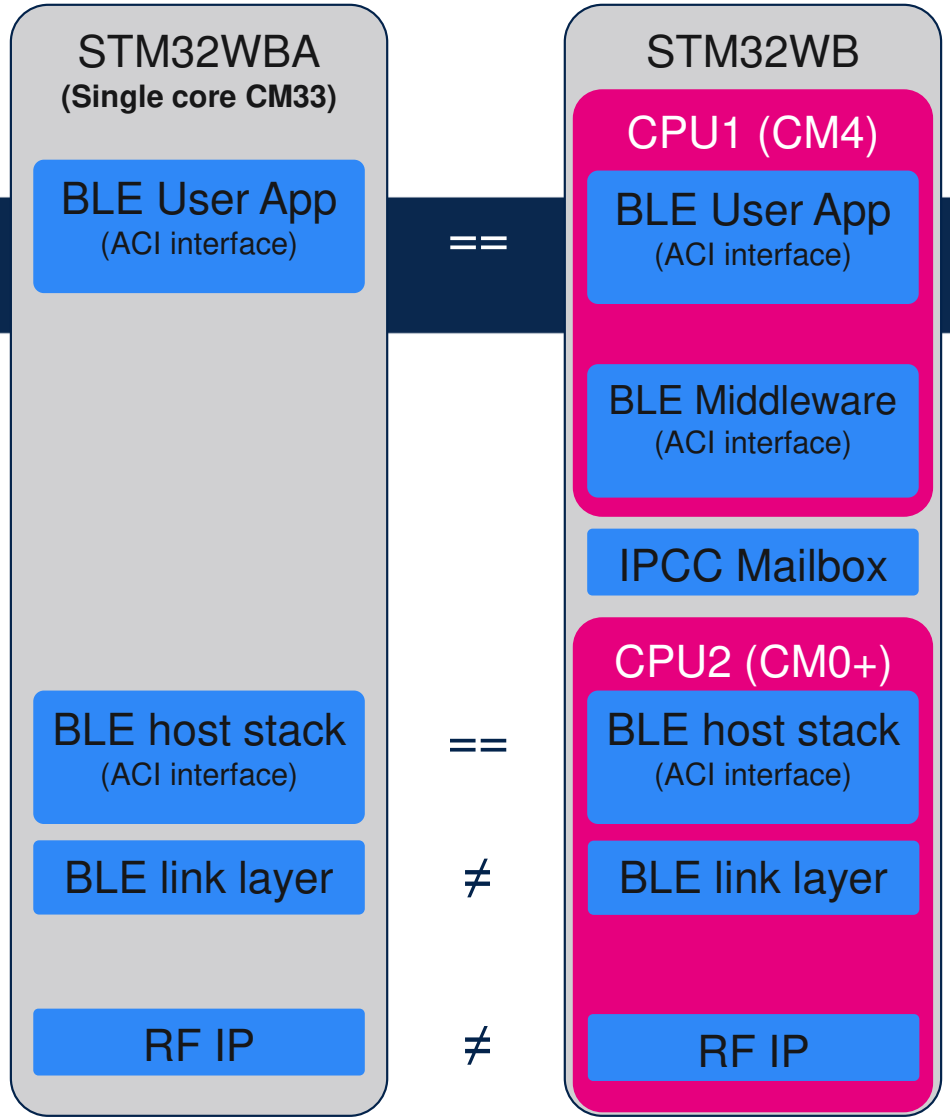
BSP
HAL

*Available on STM32WBA54/55

life.augmented

STM32WBA layers

Comparison with STM32WB



Both the STM32WB and the STM32WBA5X runs the same BLE Host Stack

= The same user ACI interface

Any BLE application based on ACI interface can be running with no modification on both product.

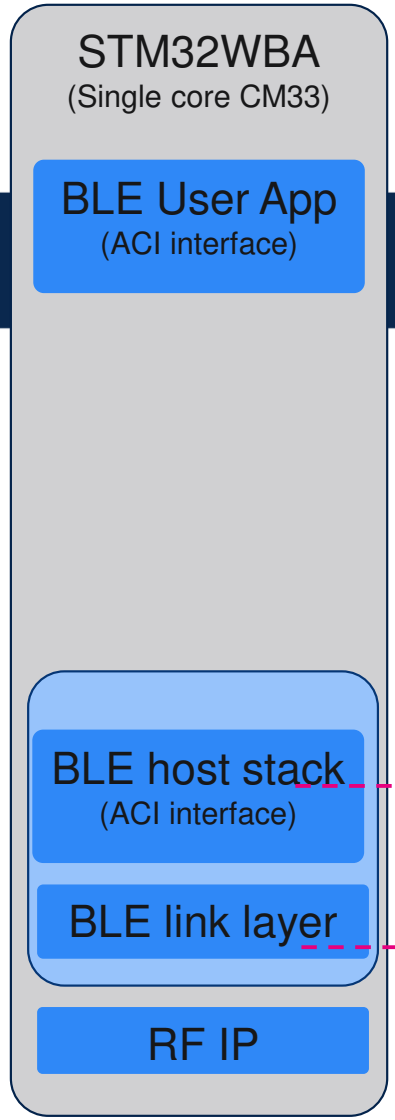
Easy porting between platforms

STM32WBA SW architecture

BLE stack delivery

STM32WBA BLE stack options

BLE stack layers



Full Host stack: stm32wba_ble_stack_full.a
Basic Host stack: stm32wba_ble_stack_basic.a
Link Layer only stack: stm32wba_ble_stack_llo.a
Link Layer only Basic stack: stm32wba_ble_stack_llobasic.a

LinkLayer_BLE_Basic_lib.a
LinkLayer_BLE_Full_lib.a

Application + Host Lib + LL lib = final binary

The stacks are delivered as certified static library

STM32WBA BLE stack options

Each BLE stack needs to use also appropriate BLE link layer library

Host stack	Full	Basic	LL_ONLY	LL_ONLY_BASIC
Link Layer required	BLE_Full_lib	BLE_Basic_lib	BLE_Full_lib	BLE_Basic_lib

- **Full stack** : all the legacy stack supported features plus the extended advertising, GATT caching, ACI HCI flow control, isochronous support for audio, L2CAP connection-oriented channels.
- **Basic stack** : no extended advertising, neither GATT caching, nor ACI HCI flow control, nor isochronous support, nor L2CAP connection-oriented channels.
- **Link Layer Only stack** (stm32wba_ble_stack_llo.a) contains all the features supported by the full stack but doesn't include the Host Stack (GATT, GAP and L2CAP features).
- **Link Layer Only Basic stack** contains all the features supported by the basic stack but doesn't include the Host Stack (GATT, GAP and L2CAP features).

BLE stack content available on associated wiki page
https://wiki.st.com/stm32mcu/wiki/Connectivity:STM32WBA_BLE_Stack_Integration

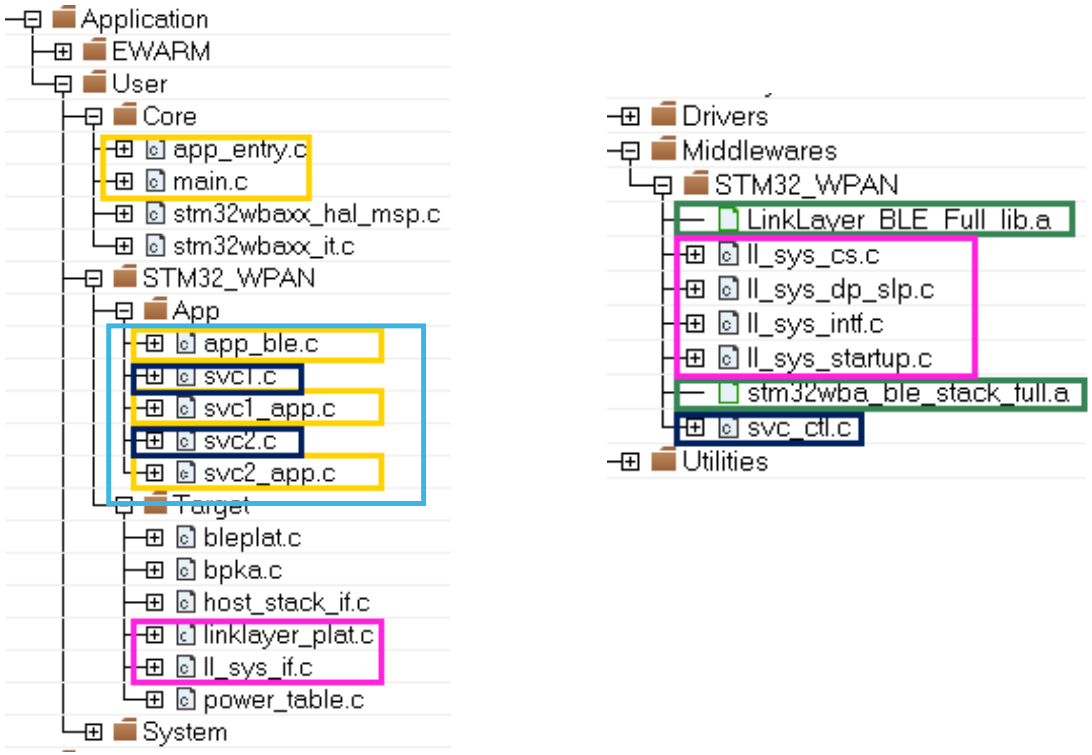
STM32WBA SW architecture

SW project organization

life.augmented

GATT Server code generated

Files in STM32WBA project generated by STM32CubeMX



Main applicative part files

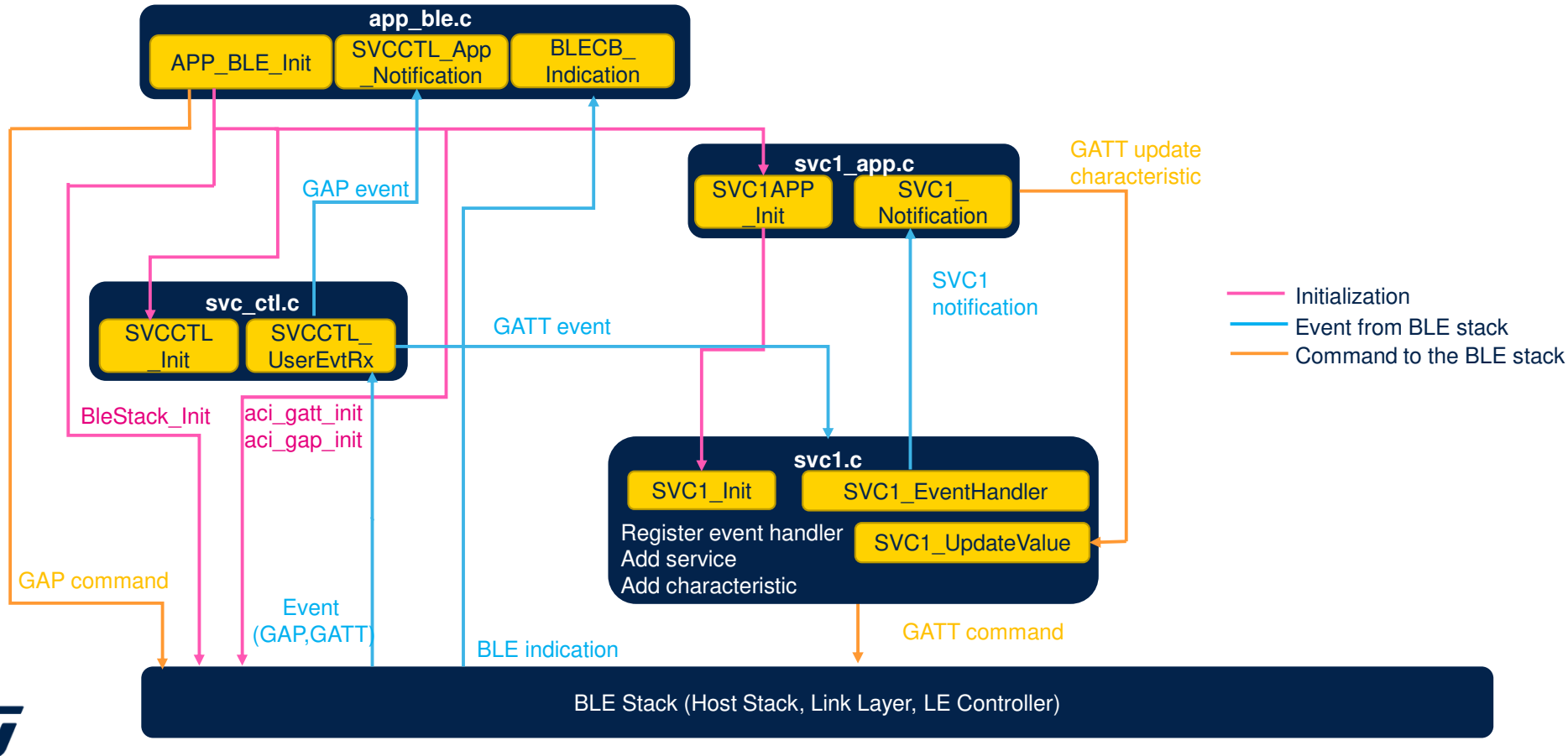
Service management

Link layer system integration files

BLE libraries

Host stack initialization files

Flows between GATT Server functions

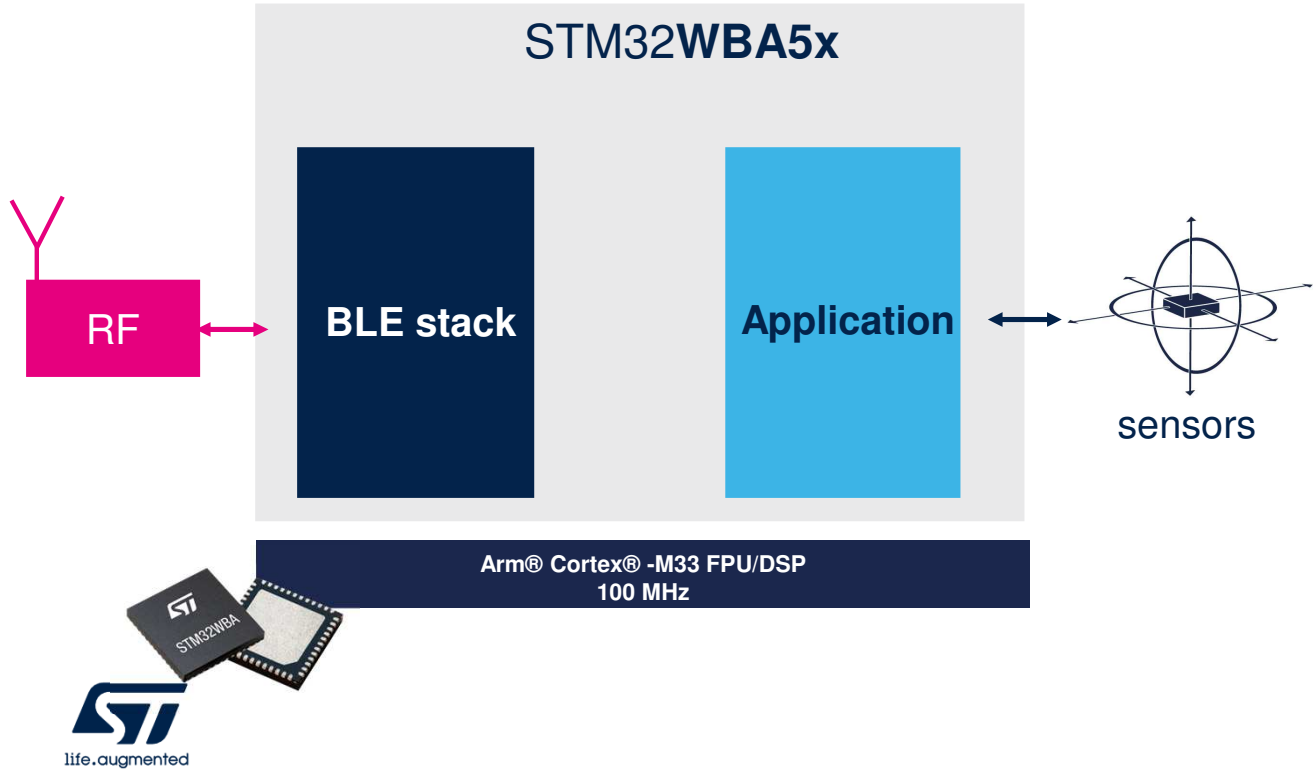


Single Core solution

life.augmented

STM32WBA RF scheduling

STM32WBA5x is single core hosting application and BLE stack



STM32WBA5x RF IP able to sustain RF activities without preempting resources of the Core

system setting need to ensure proper interrupts priority for RF handling

STM32WBA RF scheduling Interrupt priority

Interrupt name	Priority
LL_high_ISR (When the Radio is active)	0
RCC	1
User (System and Peripheral) Interrupt	2 - 4
LL_high_ISR (When the Radio is not active)	5
User (System and Peripheral) Interrupt	6 - 13
Systick	14

LL_high_ISR

ISR Process radio events . Is time critical and required highest priority

When the LL_high_ISR is not executed on time, the Radio event is skipped

only performances (latency in data transfer) on the communication with the remote device are impacted , the link is preserved

STM32WBA SW architecture

FW – intropack utilities

System Clock Manager

Managing CPU system clock source and frequency

Used by default in repository examples
Change clock setting during runtime based on application needs
Main purpose “I power optimization”

	purpose	VOS*	Flash latency	SRAMs latency
HSE 16 MHz	No radio activity	2	1	1
HSE 32 MHz	Radio activity	1	0	0
PLL	High performance (BLE audio)	1	3	0

*VOS – internal regulator Voltage Scalling

STM32CubeWBA: System Clock Manager - stm32mcu

Sequencer

Simple alternative to a real time OS

Sequencer to
ease &
secure
your SW implementation

Features

- Up to 32 tasks registered
- Request a task to be executed
- Task pause and resume
- Wait for a specific event (not blocking)
- Priority on tasks
- Allow management of an IDLE task

Does not provide preemption mechanisms
Alternative to ThreadX or FreeRTOS

Sequencer – implementation

Sequencer API

UTIL_SEQ_RegTask()	→	register a function
UTIL_SEQ_SetTask()	→	schedule task for execution
UTIL_SEQ_Run()	→	background while loop, execute scheduled task called in main while loop

Time server

Timing of events within the system

Facilities to enable application timer

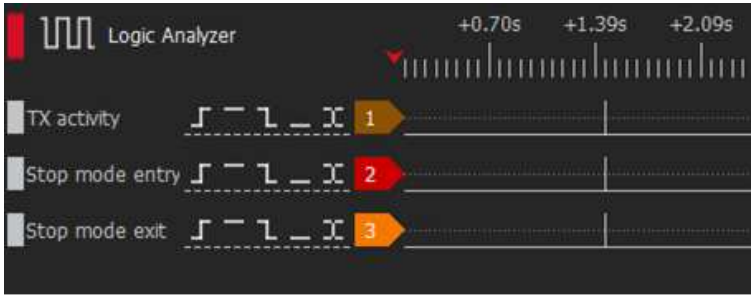
Features

- Single shot or repeated mode
- Use RTC IP (sourced by LSE)

API (not complete list)

UTIL_TIMER_Create()	initialize the timer server
UTIL_TIMER_Start() UTIL_TIMER_Stop()	start or stop timeserver
UTIL_TIMER_DeInit()	delete a virtual timeserver

GPIO real time signalization of events on low layers



Signals:

- Start/end of an interrupt
- Start/end of each a process
- Identification of specific procedures and machine states

Association of the SW signal to the desired GPIO at compilation time

Debug and traces

Examples in repository are delivered with low power focus

- Low power modes enabled
- Debug pins disabled
- LED signalization disabled
- BLE traces over UART disabled

Setting in app_conf.h:

CFG_LPM_SUPPORTED = 1

CFG_DBG_SUPPORTED = 0

CFG_LED_SUPPORTED = 0

CFG_DEBUG_APP_TRACE = 0

Q&A

life.augmented

Thank you

life.augmented