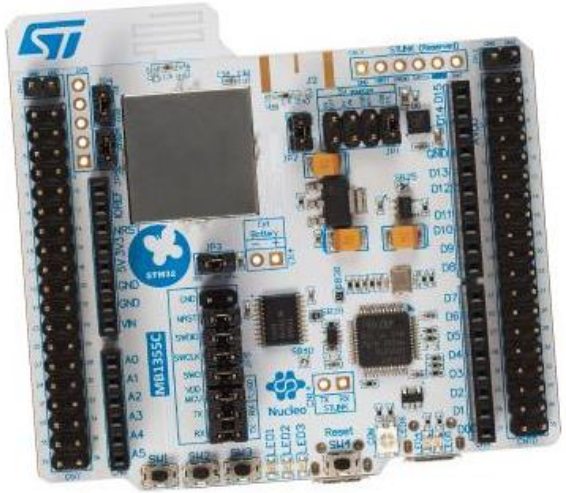
STM32WB MCU series

Excellence in **Connectivity**

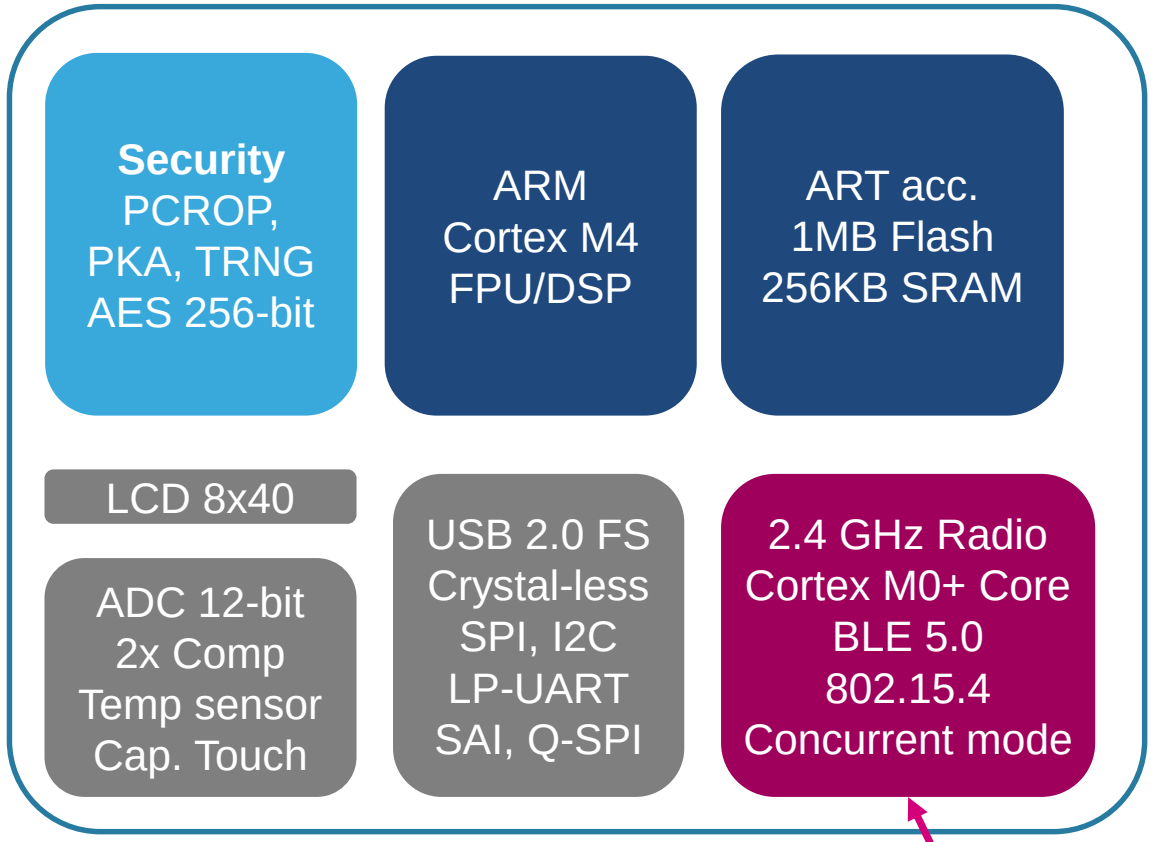
life.augmented

About the STM32WB



life.augmented

About the STM32WB

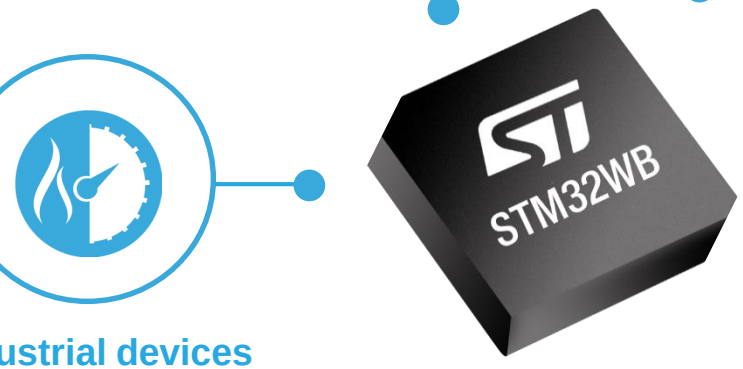


KEY FEATURES

- **2 independent cores for real time execution**
- **Ultra-low-power consumption**
  - 50 µA/MHz Active mode (at 3.0V)
  - 1.8 µA Stop mode (Radio in standby + 256KB RAM)
  - < 50 nA Shutdown mode
- **Peripherals**
  - 2xI^2C, 1xUSART, 1xLP-UART, 2xSPI, 1x USB 2.0 FS device supporting Battery Charging Detection, 1xSAI, Q-SPI (XIP), 6x 16-bit timer (including LPWM and low-power one)
- **1.71 to 3.6V voltage range (DC/DC, LDO)**
- **-40°C to +105°C temperature range**

life.augmented

All in one MCU - Full flexibility control



Lighting

- Robust RF link **-100dBm** sensitivity with IEEE 802.15.4 and **+6 dBm** output power
- Upgrade legacy 802.15.4 device to **BLE 5**
- **Update** securely Radio and stack firmware with build-in RSS
- BLE 5 and 802.15.4 protocols **Mesh capable** to extend network range

Fleet maintenance

- Retrofit legacy product to **BLE 5** and concurrency mode
- Remotely upgrade device with **OTA capability**
- **Brand protection** with Authenticated **FW upgrade** system

Industrial devices

- **Up to 105°C** radio capable
- Down to **600 nA mode with RTC** and 32KB of RAM
- Only **5µs wakeup** time over 16 wakeup lines
- PCROP, **ECC**, TRNG, **PKA**, for best design robustness
- Reduce BOM cost with **built-in LCD booster**

Fitness/ Healthcare

- **Multipoint** BLE 5 connections
- Small form factor design with CSP**100 pins**
- Battery life time care with **< 50 nA** Shutdown mode
- Dynamic Efficient **50 µA/MHz**
- Extend memory storage with **Quad-SPI**
- Handle advanced algorithm with **1 Mbyte** of Flash
- Cost optimized product with USB 2.0 **crystal-less** device

Beaconing

- **Beacon** profile available among a huge list
- **Embedded balun** to minimize design cost
- Only **5.5mA Radio TX** current to extend beacon life time
- **Up to +6 dBm** output power to get best beacon range
- < 1.8 µA Stop mode with full RAM for **battery life** optimization
- Down to 1.71 full feature capable

Home security and Audio

- **-100 dBm** sensitivity to increase area coverage
- **Customer Key Storage (CKS)** for trustable Application update
- Manage full duplex **audio** with embedded SAI
- USB FS 2.0 with Battery **Charging Detection** for remote device

life.augmented

Make the Choice of STM32WB Series

The 7 keys points to make the difference

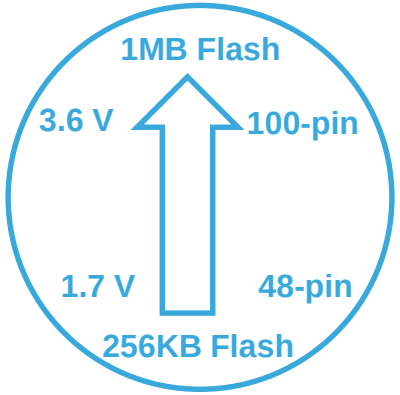
Bluetooth 5
THREAD
ZigBee

Open 2.4 GHz radio
Multi-protocol

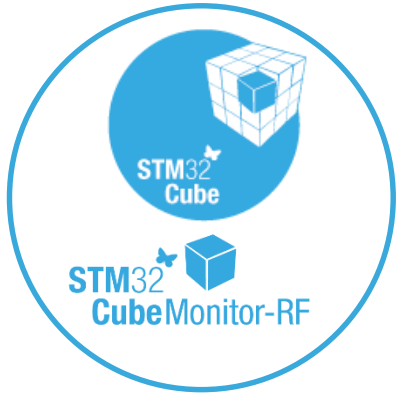
Dual-core / Full control
Ultra-low-power

IoT Protection ready

Massive integration
Cost saving



A large offer



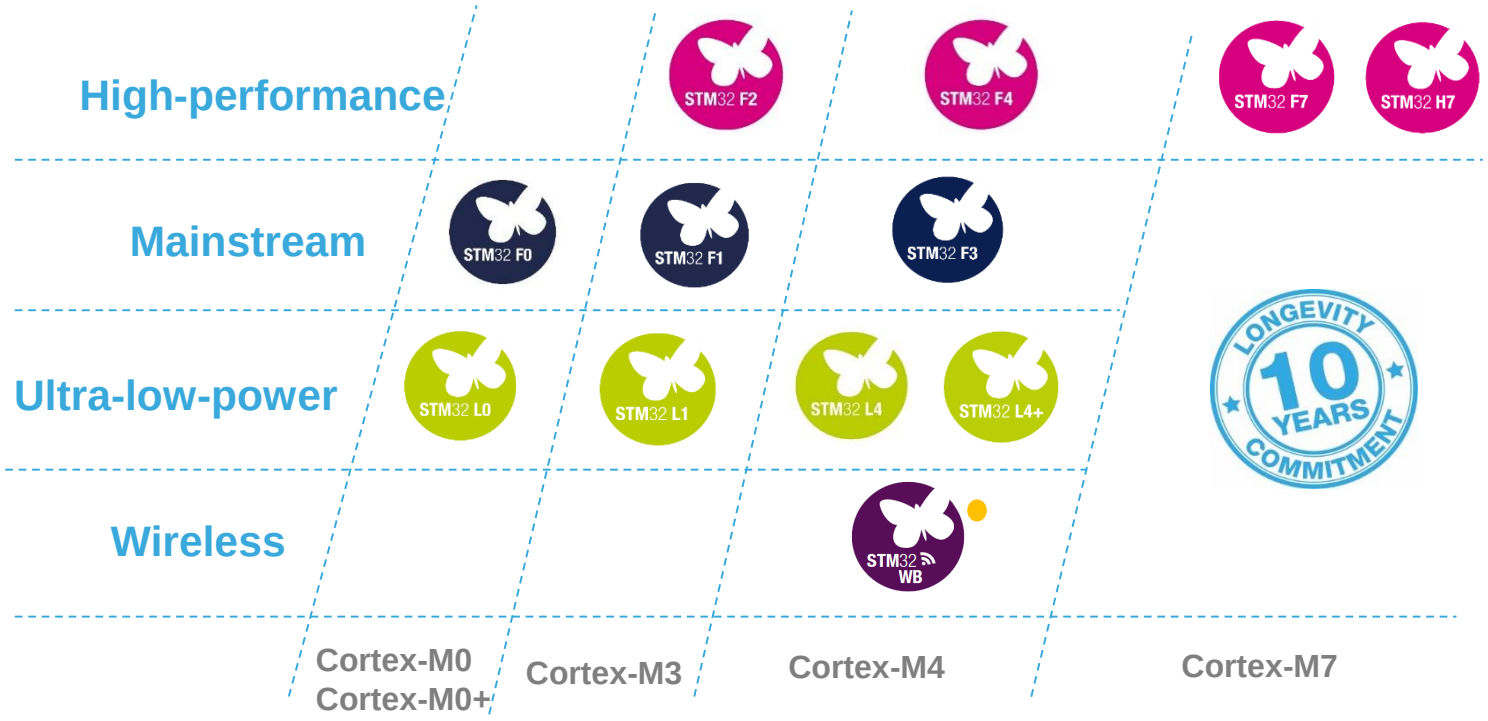
Advanced RF tool, Energy control with C code generation

No matter what!

life.augmented

Great investment

12 product series / More than 50 product lines



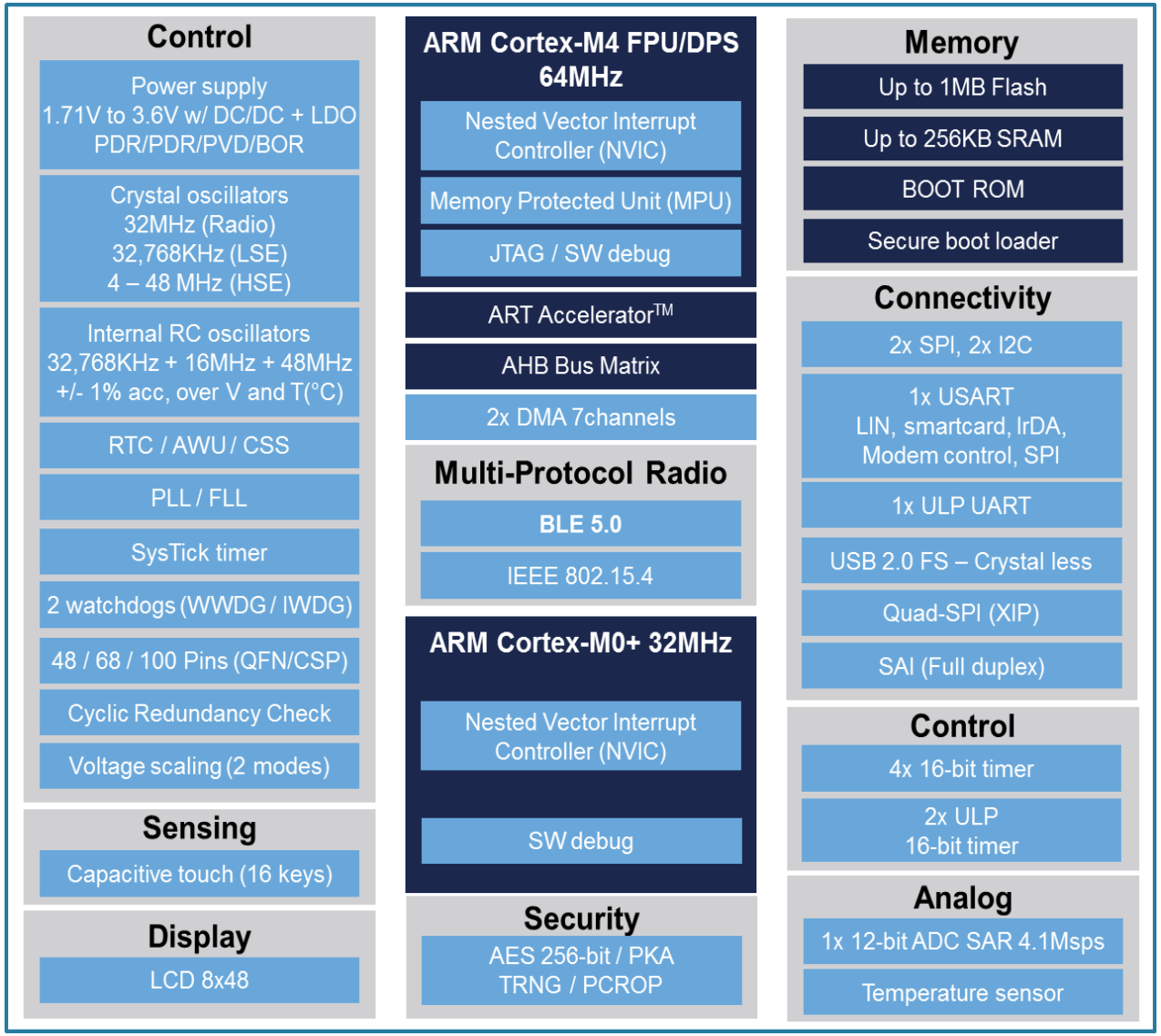
More than
40,000 customers

life.augmented

● Legend: Cortex-M0+ Radio Co-processor

STM32WB55 – Block Diagram

- CM4 DSP/FPU up to 64MHz
- CM0+ up to 32MHz
- Up to 1MB Flash and 256KB SRAM
- Radio with integrated balun
  - BT Smart **5.0** and 802.15.4
  - Output power: +6.0 dBm
  - BLE RX sensitivity: -96 dBm (-102 budget link)
  - 802.15.4 RX sensitivity: -100 dBm (-106 budget link)
  - RX: 3.8mA and TX: 5.5mA (at 0dBm)
- 1.71V to 3.6V voltage range (DC/DC, LDO)
- -40°C to +105°C temperature range
- Power consumption
  - < 53 µA/MHz Active mode (3V – RF ON)
  - 0.6 µA Standby mode (Radio in standby + 32KB RAM)
  - < 13 nA Shutdown mode



life.augmented

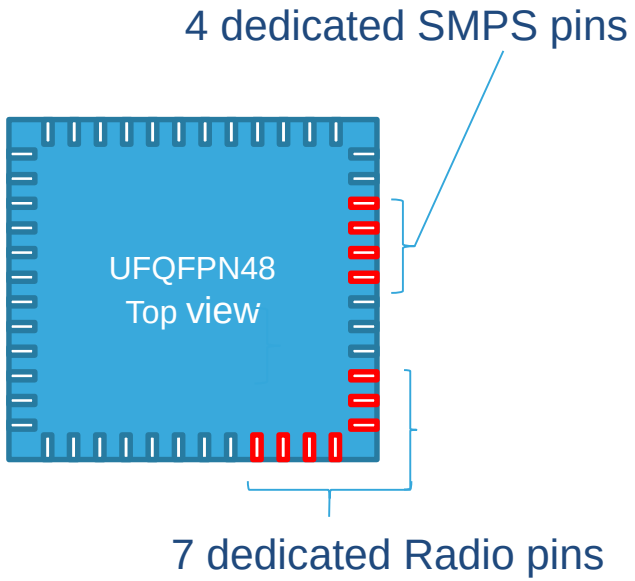
Specific STM32WB features 1/2

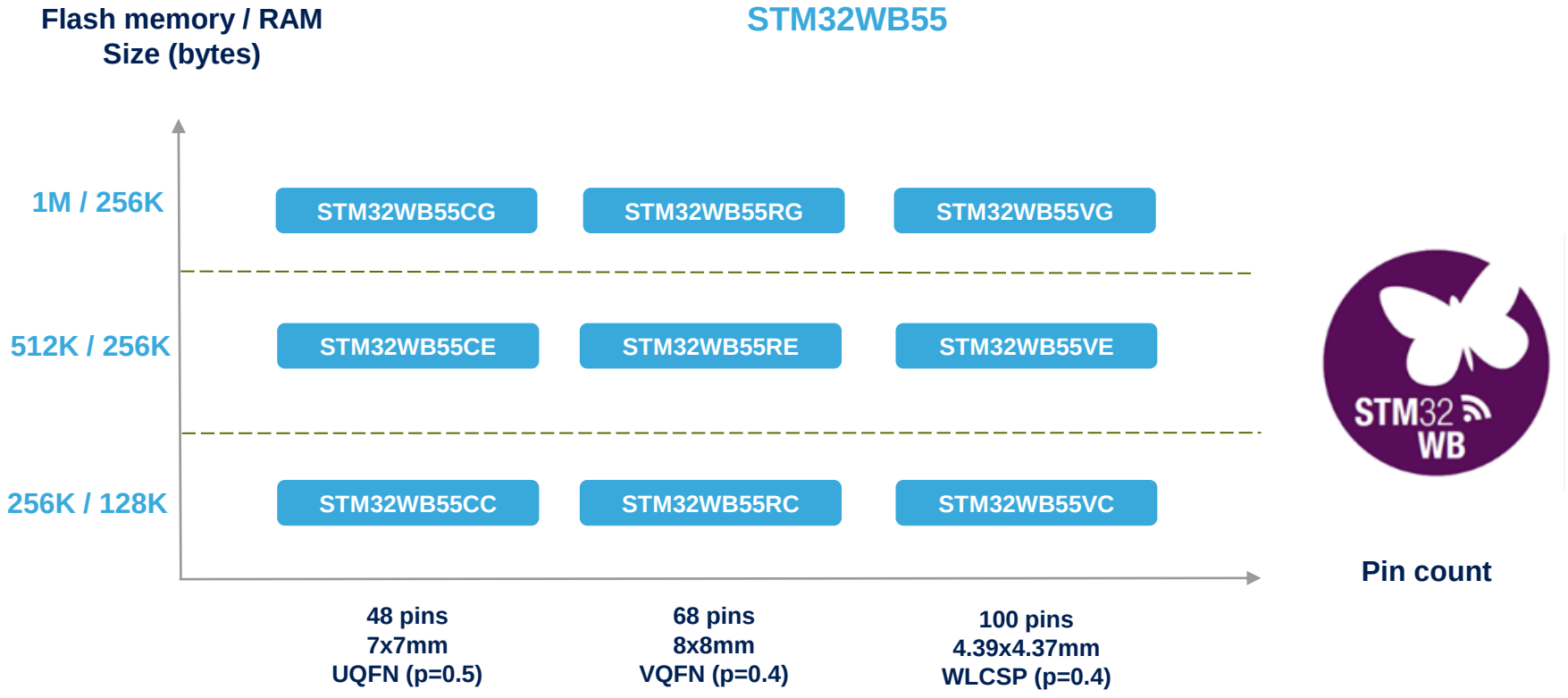
- Autonomous Radio sub-system
- Dual core system
  - Application Cortex-M4 (Boot after Reset)
  - Connectivity Cortex-M0+ (Only boot once enabled by Cortex-M4).
- Single bank shared Flash
  - On a separate AHB bus, with it’s own clock divider.
- Cortex-M0+ memory and peripheral security
- HSE fixed frequency at 32 MHz.
- 2 LSIs
  - LSI1 standard STM32 low-power LSI
  - LSI2 low-drift Radio LSI (mandatory selected for the radio sub-system)

Specific STM32WB features 2/2

- SRAM2 split in 2 parts
  - SRAM2a backup RAM
  - SRAM2b non-backup RAM
- Debug Cross trigger unit including trigger in/out.
- Integrated SMPS to supply Core and Radio LDO’s.

STM32WB series portfolio





Low power modes

Wake-up time		
	RUN (Range1) at 64 MHz	117 (73) µA / MHz**
	RUN (Range2) at 16 MHz	109 µA / MHz**
	LPRUN at 2 MHz	103 µA / MHz**
9 cycles	SLEEP at 64 MHz	41 µA / MHz
9 cycles	LPSLEEP at 2 MHz	45 µA / MHz
1.7 µs	STOP 0 (full retention)	100 µA
4 µs (19 µs)	STOP 1 (full retention)	9.2 µA / 9.6 µA*
5 µs (20 µs)	STOP 2 (full retention)	1.8 µA / 2.2 µA*
14 µs (25 µs)	STANDBY + 32 KB RAM	320 nA / 650 nA*
14 µs (25 µs)	STANDBY	110 nA / 440 nA*
50 µs	SHUTDOWN	30 nA / 315 nA*
	VBAT	2 nA / 300 nA*

RF Operation

CoreMark score	ULPBench score
216	136

www.eembc.org

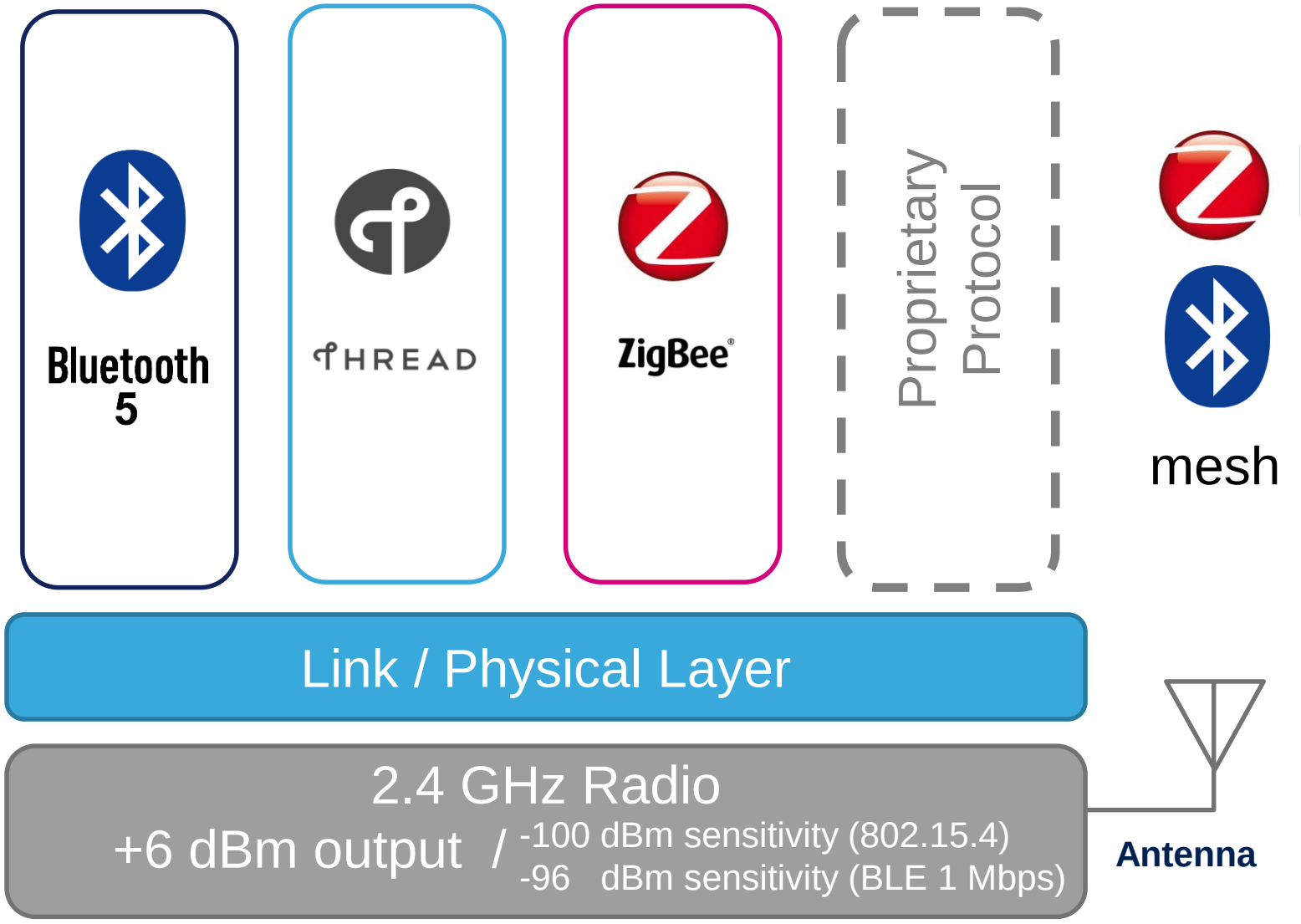
Typ @ VDD =1.8 V @ 25 °C

Typ @ V_{DD} = 1.8 V @ 25 °C

** : with RTC*
*** : from SRAM1*

() Typ. value for SMPS mode

Wireless part - make it Yours



Overview

- Discover our ultra-low-power wireless communication STM3255 Nucleo pack
  - 2-board kit based on the STM32WB microcontroller
    - USB dongle
    - STM32WB55 Nucleo board
  - Enables a wide diversity of applications
  - Comes with various packaged software examples

Application benefits

- Designed for low-power, wireless communication use cases
- Turnkey demonstration firmware
- Develop your own application

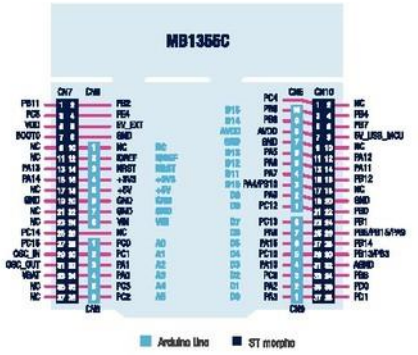
life.augmented

STM32WB55 Nucleo Pack insert card

STM32WB55 Nucleo Pack

Wireless and ultra-low-power BLE 5.0 & IEEE 802.15.4

Preliminary version



NUCLEO BOARD FEATURES

- **STM32WB55RGV6 MCU**
- PCB antenna and SMA connectors
- Flexible power supply supporting CR2032
- Dual USB Port (Application/debug)
- Multiple USER switch/LEDs
- Supports Arduino™ and ST morpho connectors
- Embedded ST-LINK/V2-1 debugger and programmer
- Arm® Mbed Enabled™

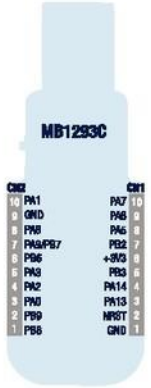
USB DONGLE

- **STM32WB55CGU6 MCU**
- PCB antenna and UFL connectors
- Cuttable PCB

STM32WB55 FEATURES

- Arm® Cortex®-M4 MCU with **64 MHz/80 DMIPS** and dedicated Arm® Cortex®-M0+ radio co-processor at **32 MHz**
- **Multiprotocol:** BLE 5.0, 802.15.4, concurrent mode
- 1 Mbyte of Flash memory
- 256 Kbytes of SRAM
- **High RF performances :** RX -96 dBm/ -100 dBm; TX +6 dBm

STM32 Nucleo



Download on the App Store

GET IT ON Google Play

ST part number: P-NUCLEO-WB55

www.st.com/x-nucleo

Wireless and ultra-low-power STM32 Nucleo Pack based on STM32WB

GETTING STARTED

1/ Plug the Nucleo USB_STLINK connector (P2P Server) and USB dongle (P2P Client) to power sources. On the P2P server, you can see LED-blinking showing it is advertising during 1 minute.

2/ Once the P2P client is powered, push the SW1 button to start scanning (Blue LED ON 5s); It then connects automatically to the P2P server.

3/ Once connected, the green LED is blinking for each connection interval. The P2P client searches for the P2P service, LEDs & buttons characteristics, and enables notification.

4/ Pushing the SW1 button toggles the blue LED on the remote device.

5/ Pushing the SW2 button on the Nucleo board changes the connection Interval (50 ms, 1 s). The effect is visible directly on the green LED of the Nucleo board.

6/ The demo software and several software examples that allow you to use the STM32 Nucleo and USB dongle features are available at www.st.com/stm32nucleo

7/ Develop your own applications using available examples.

SYSTEM REQUIREMENTS

- Windows® OS (7, 8,10), Linux® 64-bit or macOS®
- USB Type-A to Micro-B cable

DEVELOPMENT TOOLCHAINS

- Keil® MDK-ARM[1]
- IAR™ EWARM[1]
- GCC-based IDEs
- Arm® Mbed™ online

EMBEDDED SOFTWARE

STM32CubeWB MCU Package including LL & HAL drivers, BLE & Thread libraries, RTOS, USB & Touch sensing

SOFTWARE TOOLS

- STM32CubeMX
- STM32CubeMonRF
- STM32CubeProg

Notes:

1. On Windows® only

STANDARD PROTOCOL

THREAD

Bluetooth 5

By using or installing (as applicable) this evaluation kit you accept all the terms of the EVALUATION PRODUCT LICENCE AGREEMENT available at www.st.com/epla

APPLICATIONS STORE

ST BLE Profiles | ST BLE StarNet | ST BLE Sensor

STM32 CubeMonitor-RF

STM32 CubeProgrammer

CAN ICES-3(B)/NMB-3(B)

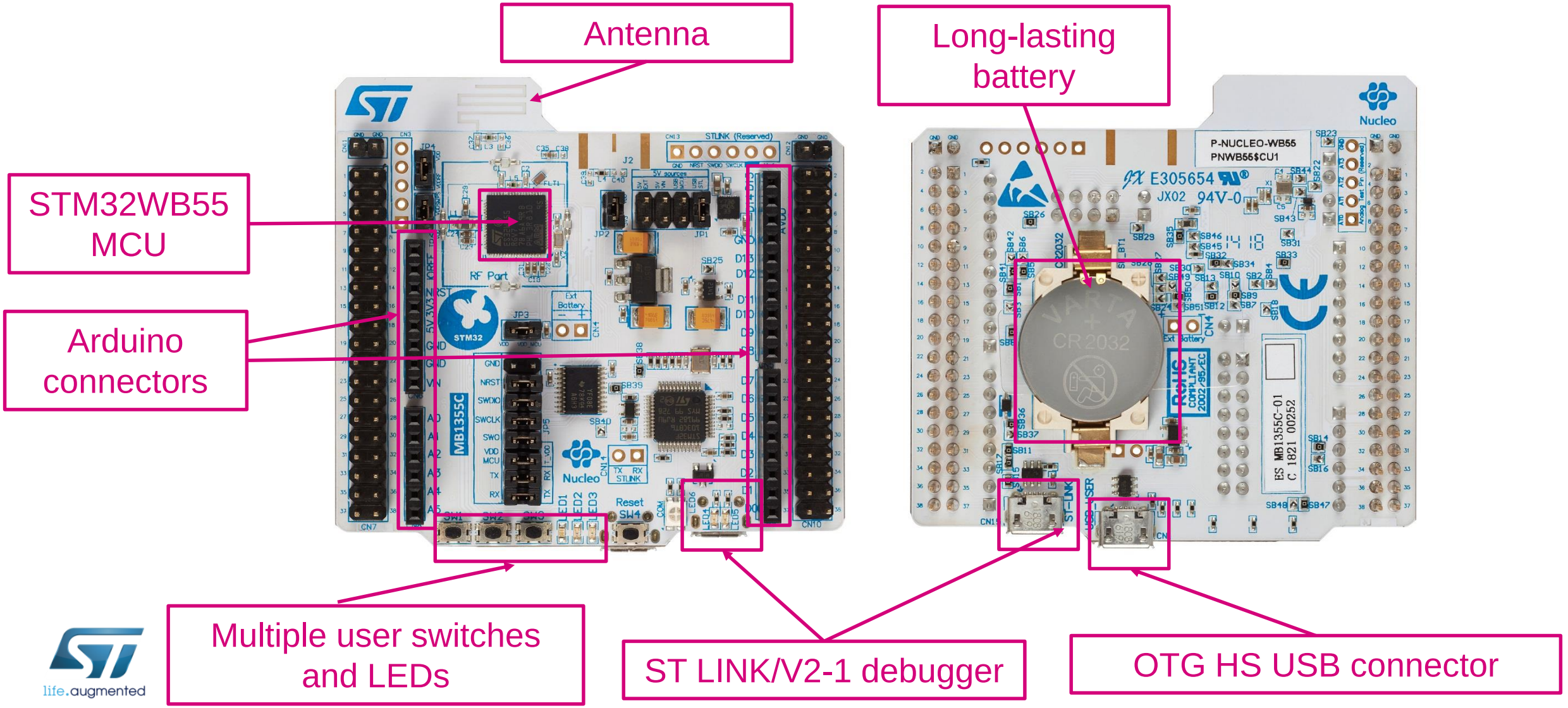
RoHS compliant

arm MBED Enabled

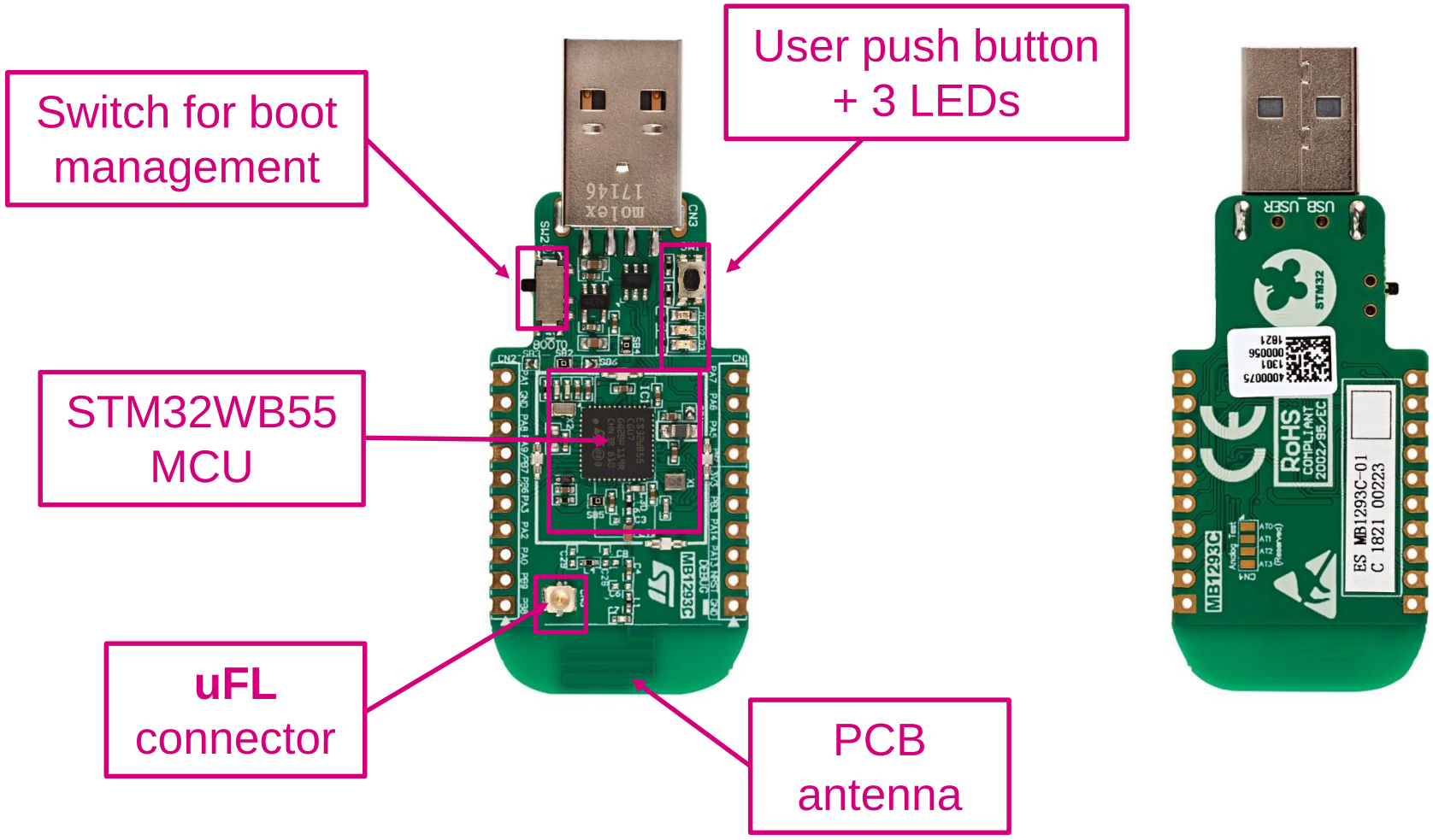
EPNWB55$CU3

Order code: P-NUCLEO-WB55

STM32WB Nucleo Board



STM32WB Dongle



Key features

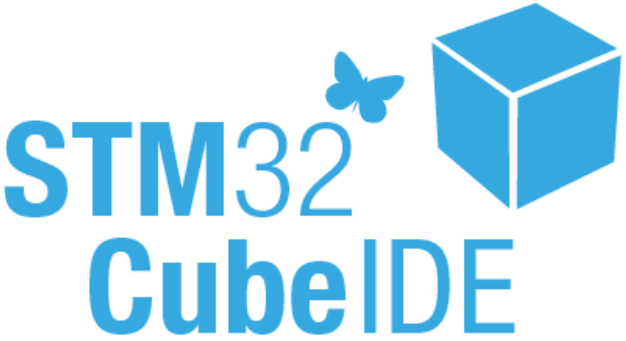
- STM32WB55 microcontroller on both boards
  - Arm® Cortex®-M4 core @ 64 MHz / 80 DMIPS
  - Arm® Cortex®-M0+ core @ 32 MHz
  - 1 Mbyte of Flash memory + 256 Kbytes of RAM
  - Support multiple wireless protocols: BLE 5.0 and IEEE 802.15.4 Concurrent mode
  - High RF performance:
    - BLE RX sensitivity: -96 dBm
    - 802.15.4 RX sensitivity: -100 dBm
    - TX: + 6 dBm
- The Bluetooth range on STM32WB55 microcontrollers is intended to be 10 m in normal working conditions, and up to 100 m in an open field with up to 8 simultaneous connections.

Demonstration SW

- Demonstration software available with the STM32WB55 Nucleo pack.
  - Based on the BlueNRG-MS software stack
  - Supports multiple roles simultaneously, and can act at the same time as Bluetooth LE sensor and hub device.
  - Supports the main adopted GATT-based profiles
  - Main application runs on the Arm® Cortex®-M4 core while the RF wireless stack runs on the Arm® Cortex®-M0+core.

- Latest versions of the demonstration source code are available at www.st.com/stm32nucleo .

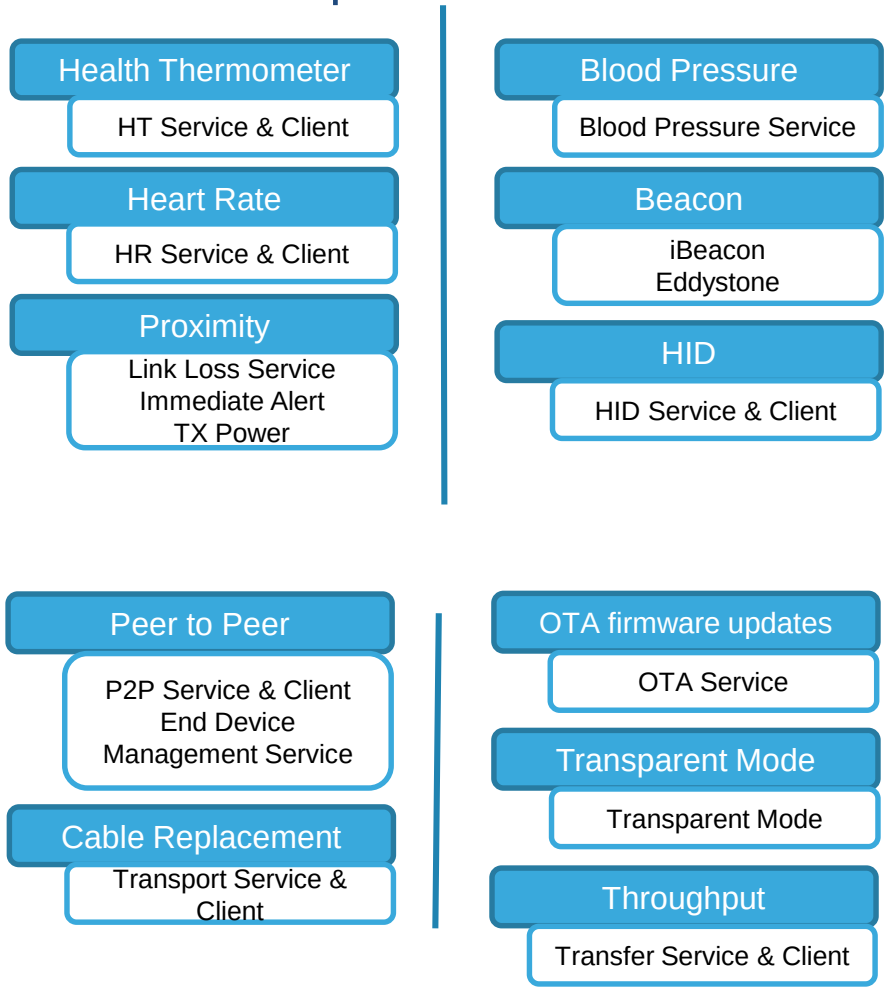
- Comprehensive free software libraries and additional examples available with the STM32Cube package



- Support for a wide range of integrated development environments (IDE) including IAR™, Keil and GCC-based IDEs like Atollic® TrueSTUDIO®

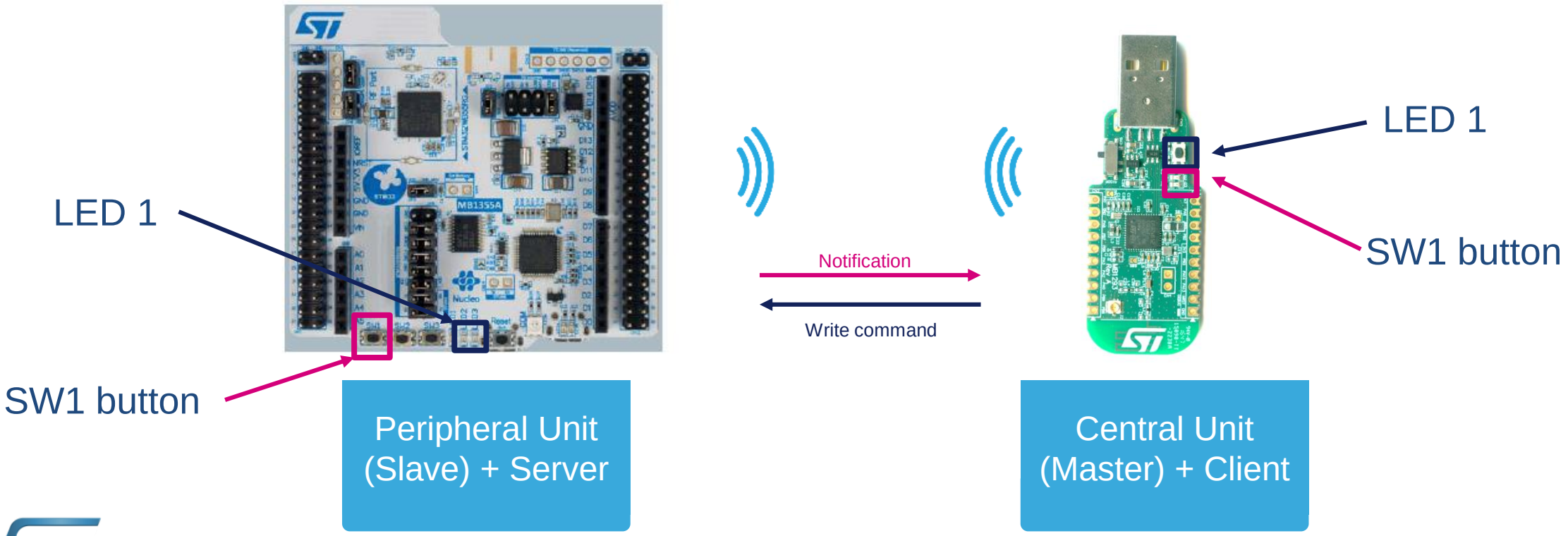
Purpose(1/2)

- This presentation summarizes the GATT-based applications developed for STM32WB and supported in the STM32Cube firmware library.
- **BT SIG GATT-based applications**
  - Beacon
  - Blood Pressure
  - Health Thermometer
  - Heart Rate
  - Human Interface Device (HID)
  - Proximity
- **Proprietary GATT-based applications**
  - Cable Replacement
  - Data Throughput
  - P2P Server – P2P Client – P2P Router
  - Over-the-air (OTA) firmware updates
  - Transparent Mode – Direct Test Mode



STM32WB P2P Demo (Out Of Box)

- Peer-to-Peer connections and data exchanges between two STM32WB MCUs
  - One server device, which broadcasts
  - One client device, which scans for services and characteristics



Note: The server and the client can be a Nucleo Board or a USB Dongle

Smartphone P2P Demo

- Peer-to-Peer connections and data exchanges between an STM32WB MCU and a Smartphone Application
  - One server device, which broadcasts
  - One smartphone, acting as a client

Download on the App Store

GET IT ON Google Play

SW1 button

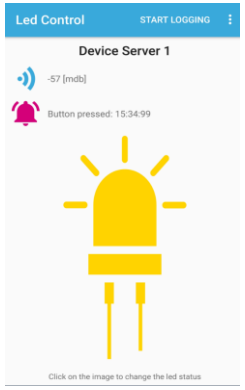
Peripheral Unit (Slave) + Server

Notification

Write command

Central Unit (Master) + Client

Use Your ST SensNet App



Virtual LED status

Note: The server can be a Nucleo Board or a USB Dongle

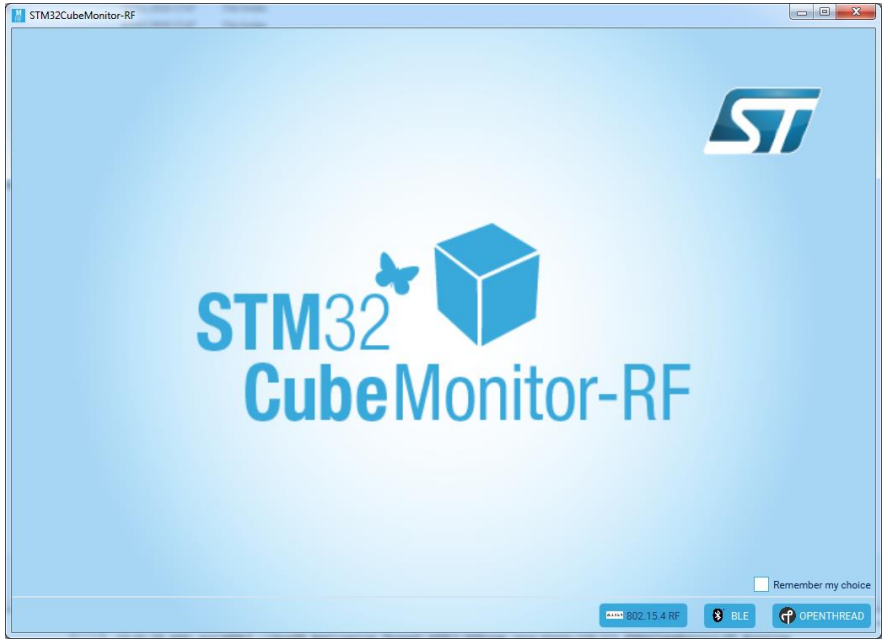
life.augmented

Reference material

- Refer to www.st.com
  - Ordering information
  - Getting started manual, user manual and application notes
  - Board schematics
  - Application development environment support
  - Demonstration firmware sources

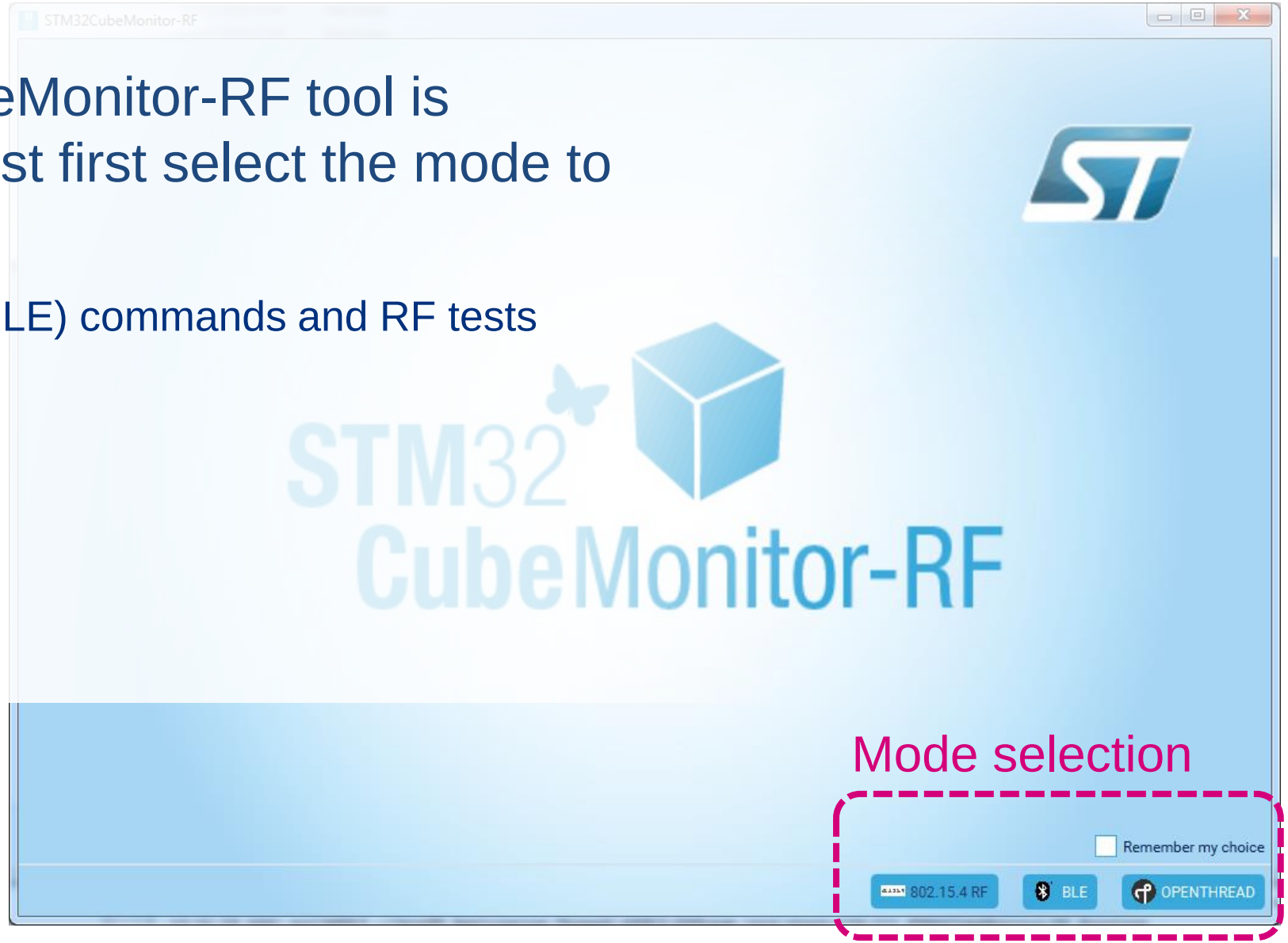
STM32CubeMonitor-RF

- The STM32CubeMonitor-RF tool is used to manage the wireless features of the STM32WB series via Bluetooth Low Energy (BLE) commands, BLE RF tests, OpenThread commands and 802.15.4 RF tests.
- Target board can be an ST Nucleo board, a USB BLE dongle or any customer board.
- The USB connection uses a Virtual COM port
- A UART port can also be used



Mode selection

- When the STM32CubeMonitor-RF tool is launched, the user must first select the mode to be used:
  - Bluetooth Low Energy (BLE) commands and RF tests
  - OpenThread commands
  - IEEE 802.15.4 RF tests.



BLE Mode

- With STM32CubeMonitor-RF in BLE mode, it is possible to:
  - Send ACI commands to the target board. The ACI commands are BLE commands used to configure and communicate with BLE peripherals. The tool provides the list of the available commands and parameter descriptions.
  - Perform RF tests (transmit, receive)
  - Flash firmware over the air (OTA)
  - Configure beacons
  - Configure advertising
  - Discover remote device services

RF test with two devices

Connect the STM32WB device

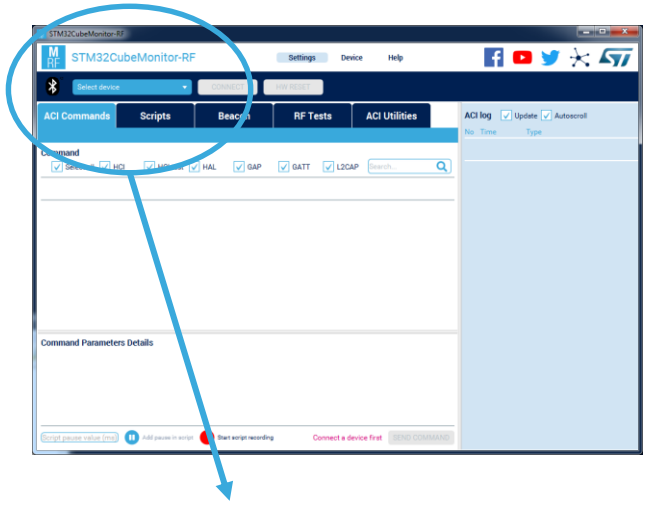
- STM32CubeMonitor-RF supports both the board and the USB dongle of the STM32WB55 Nucleo pack.
- The “Transparent Mode” firmware has to be loaded on the target device.
- Connection instructions:

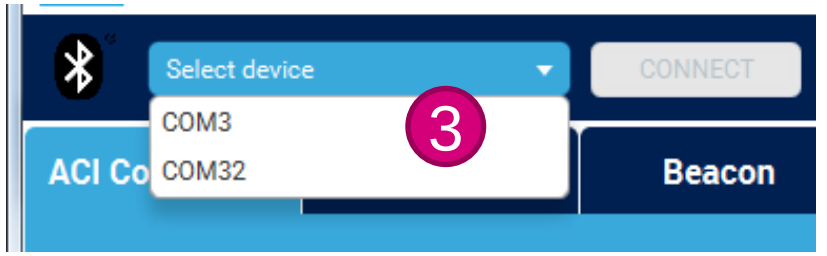
  1) Connect the device to the USB (or UART) port

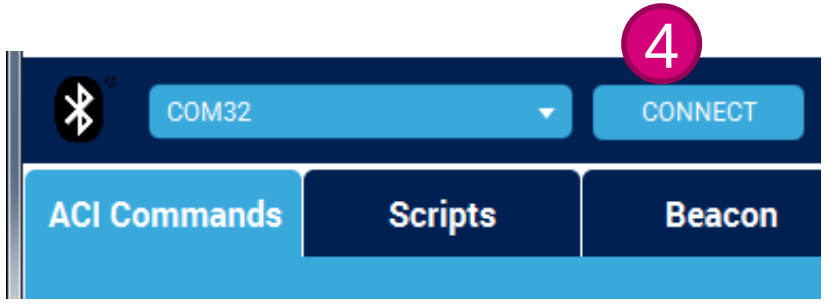
  2) Wait for the COM port detection

  3) Click “Select device” to select the relevant COM port

  4) Press “Open” to start communication



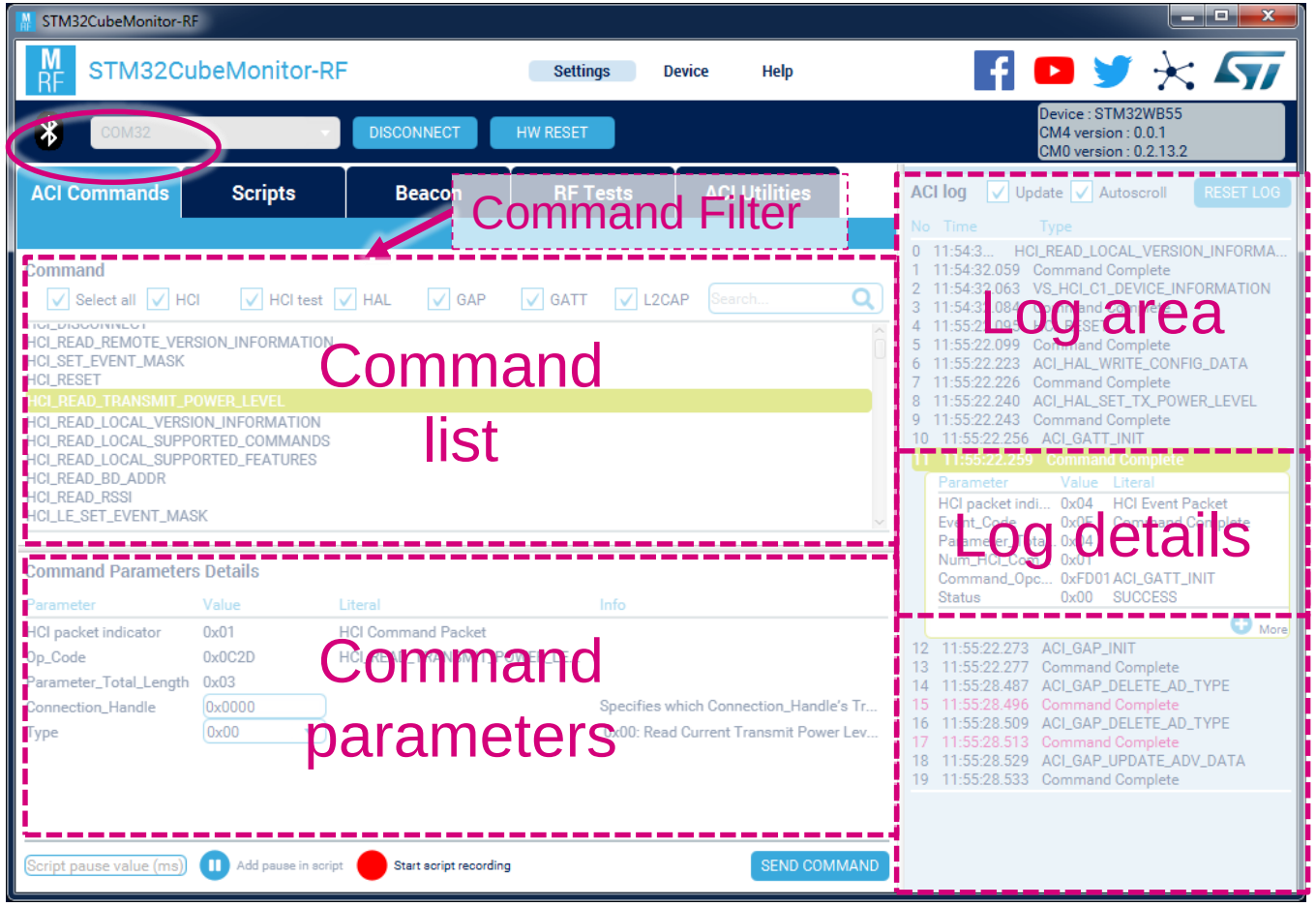




life.augmented

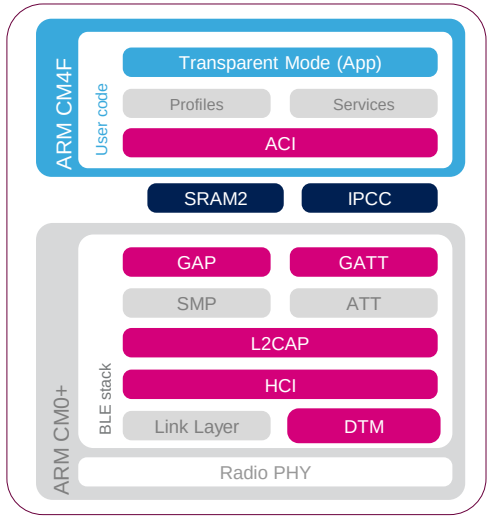
ACI command panel

- This panel is used to prepare ACI commands that will be sent to the target device.



Command categories:

- HCI
- HCI test
- HAL
- GAP
- GATT
- L2CAP



life.augmented

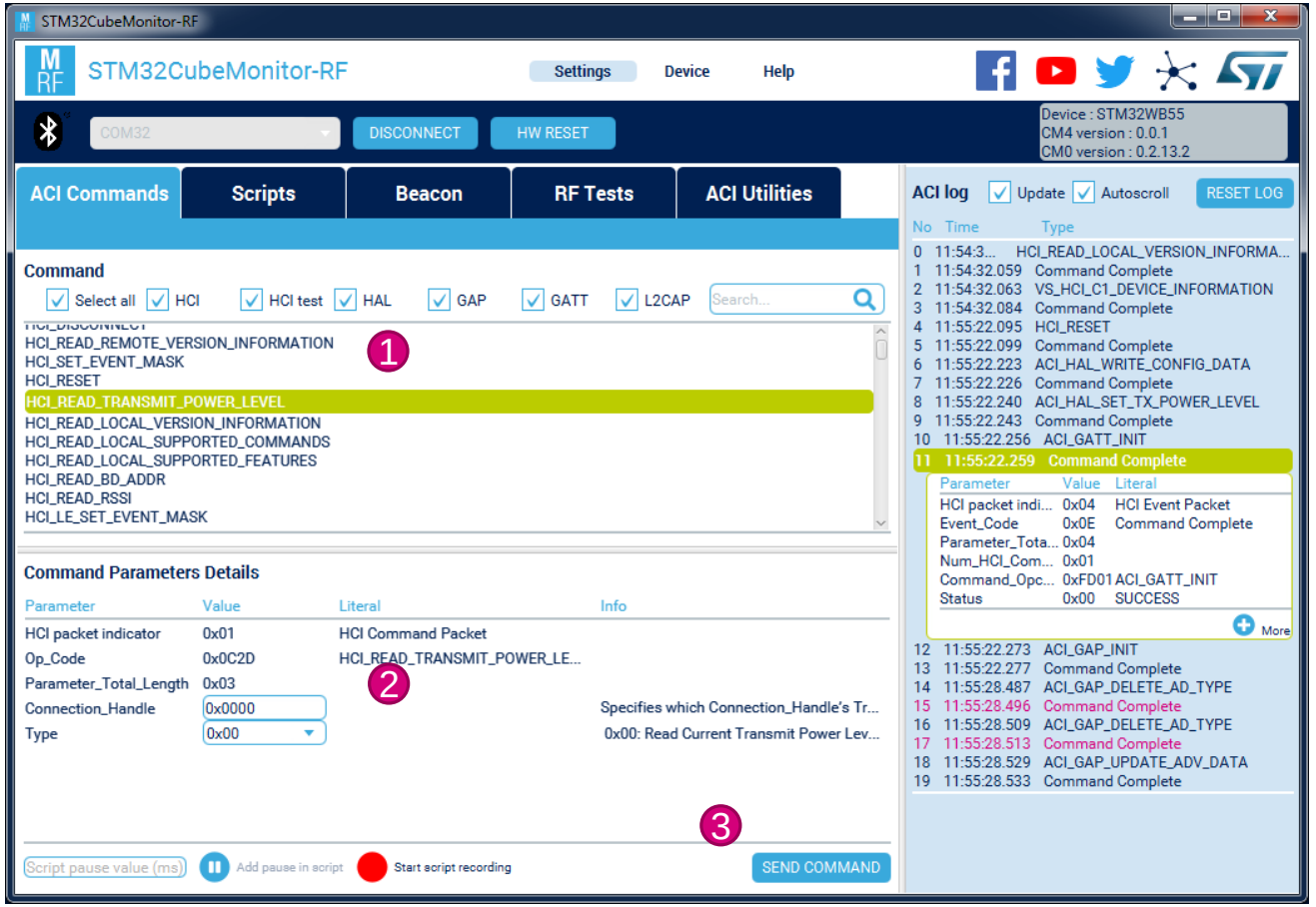
Send one ACI command

1. Select an ACI command from the command list
2. Edit relevant parameters
3. Press “Send”

- The ACI commands and ACI responses are displayed in the “log area”.

Tips :

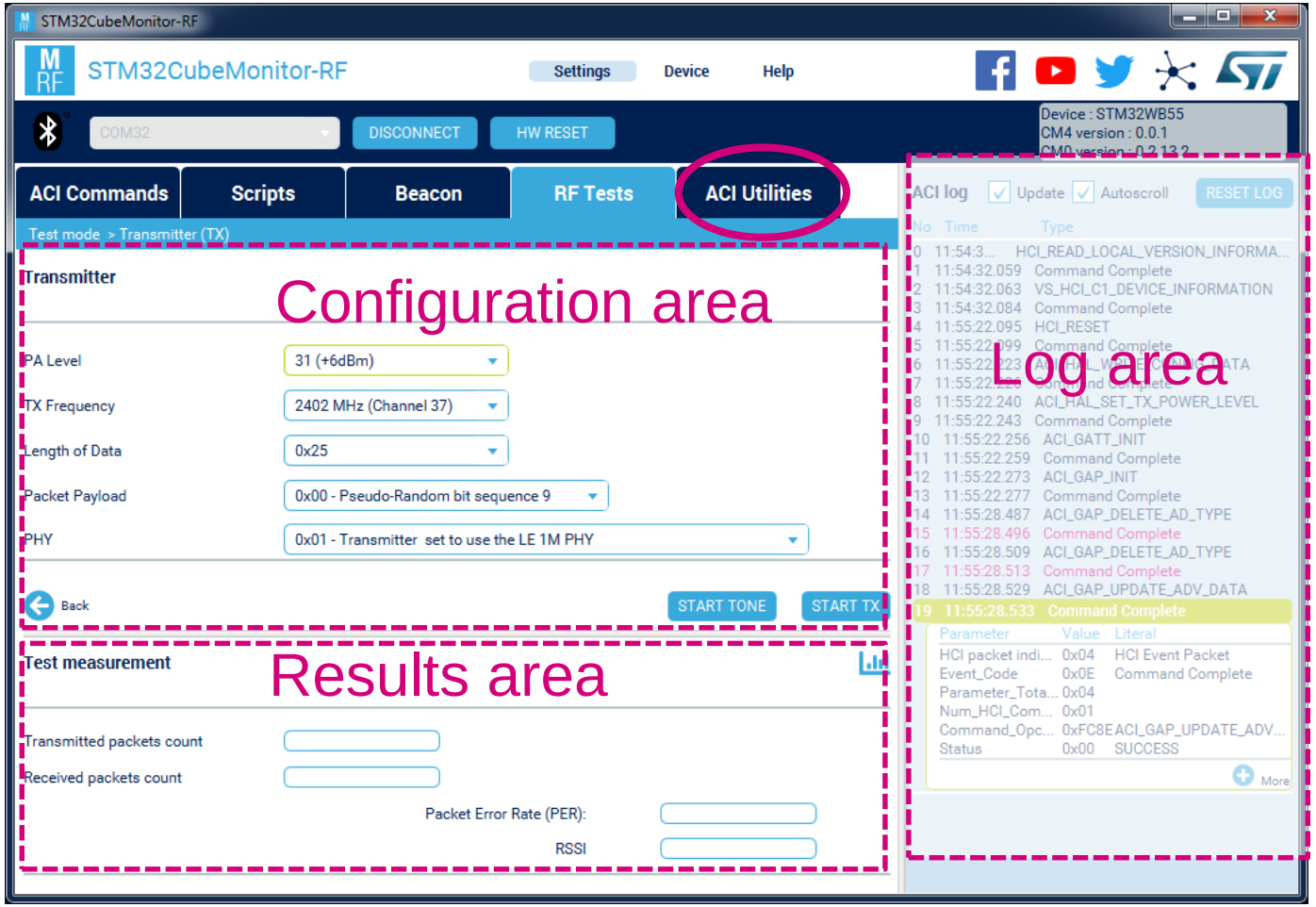
- When a command fails, the result is highlighted in pink.
- Double click on the command in the log area to display its raw data and details in a pop-up window



life.augmented

RF Test panel

- The RF Tests tab can perform the following tests:
  - Test packet transmission
  - Test packet reception
  - Start/Stop a tone signal
  - Perform BLE Packet Error Rate (PER) measurement



Select the RF test

The first step is to select the RF test mode:

- Transmitter test (TX)
- Packet error test (PER)
- Receiver Test (RX)

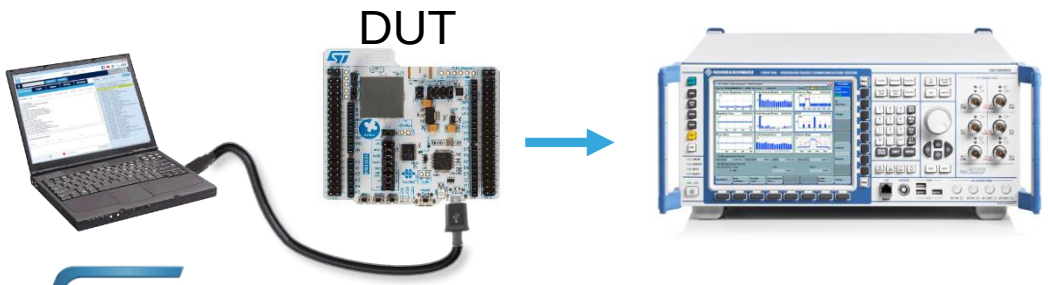
1. Select the test mode
2. Click “SELECT TEST MODE”

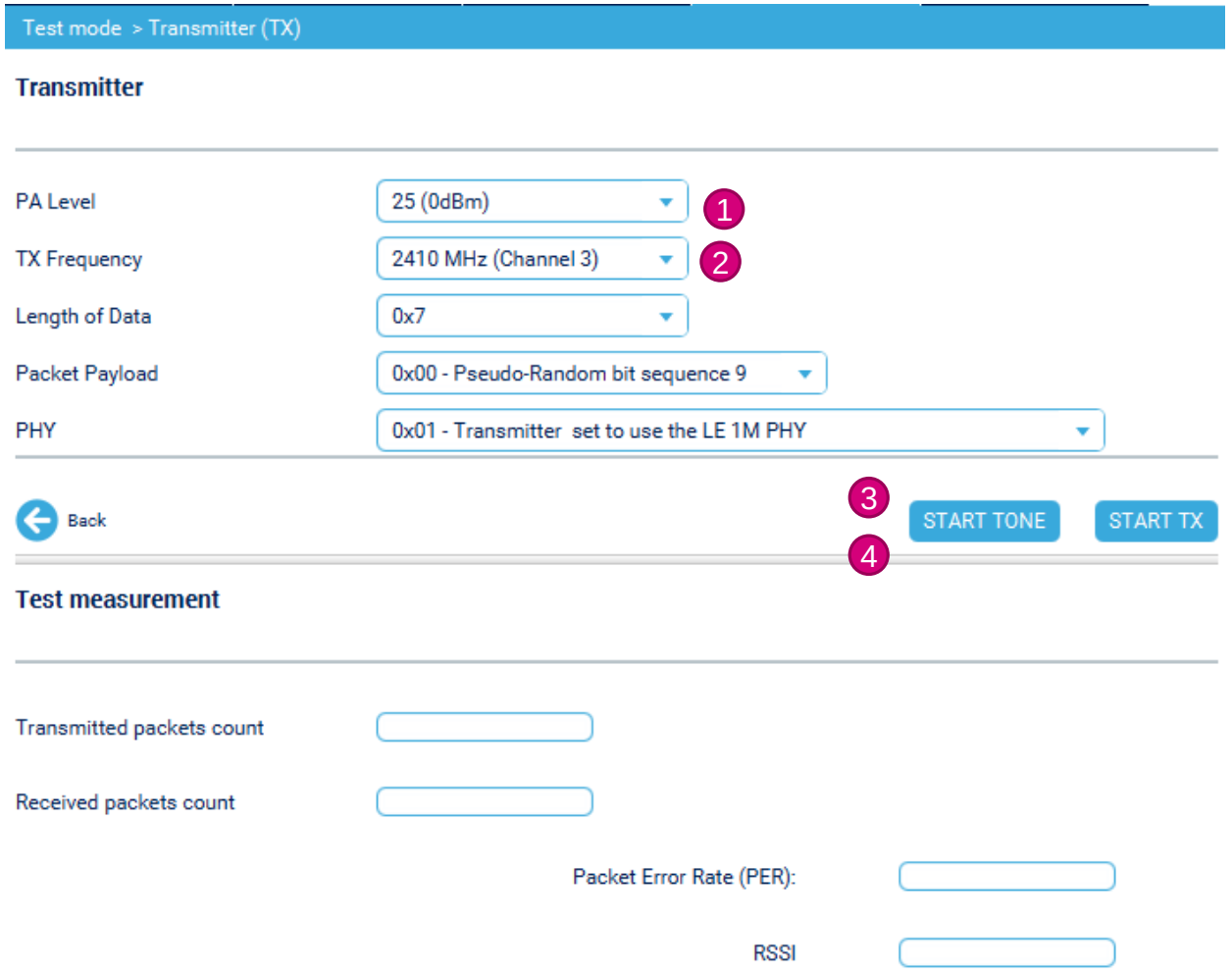
The test panel is displayed.

The test mode is described in the bar above the test configuration zone

Start/Stop the tone test

- To start the tone test:
  1. Set the transmission (TX) power.
  2. Set the TX frequency to select a specific BLE RF channel
  3. Click “START TONE”
- To stop the tone test:
  4. Click “STOP TONE”

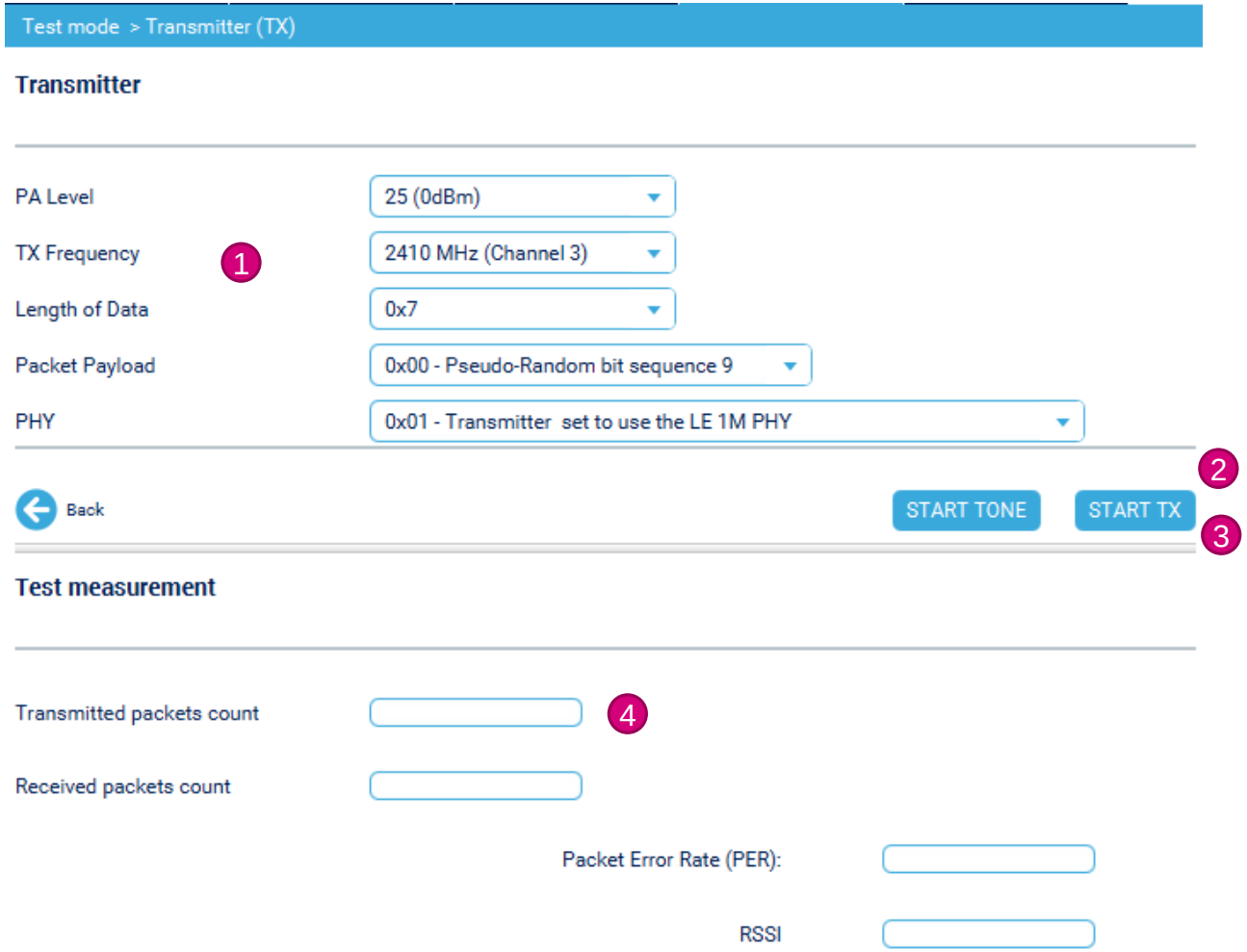




life.augmented

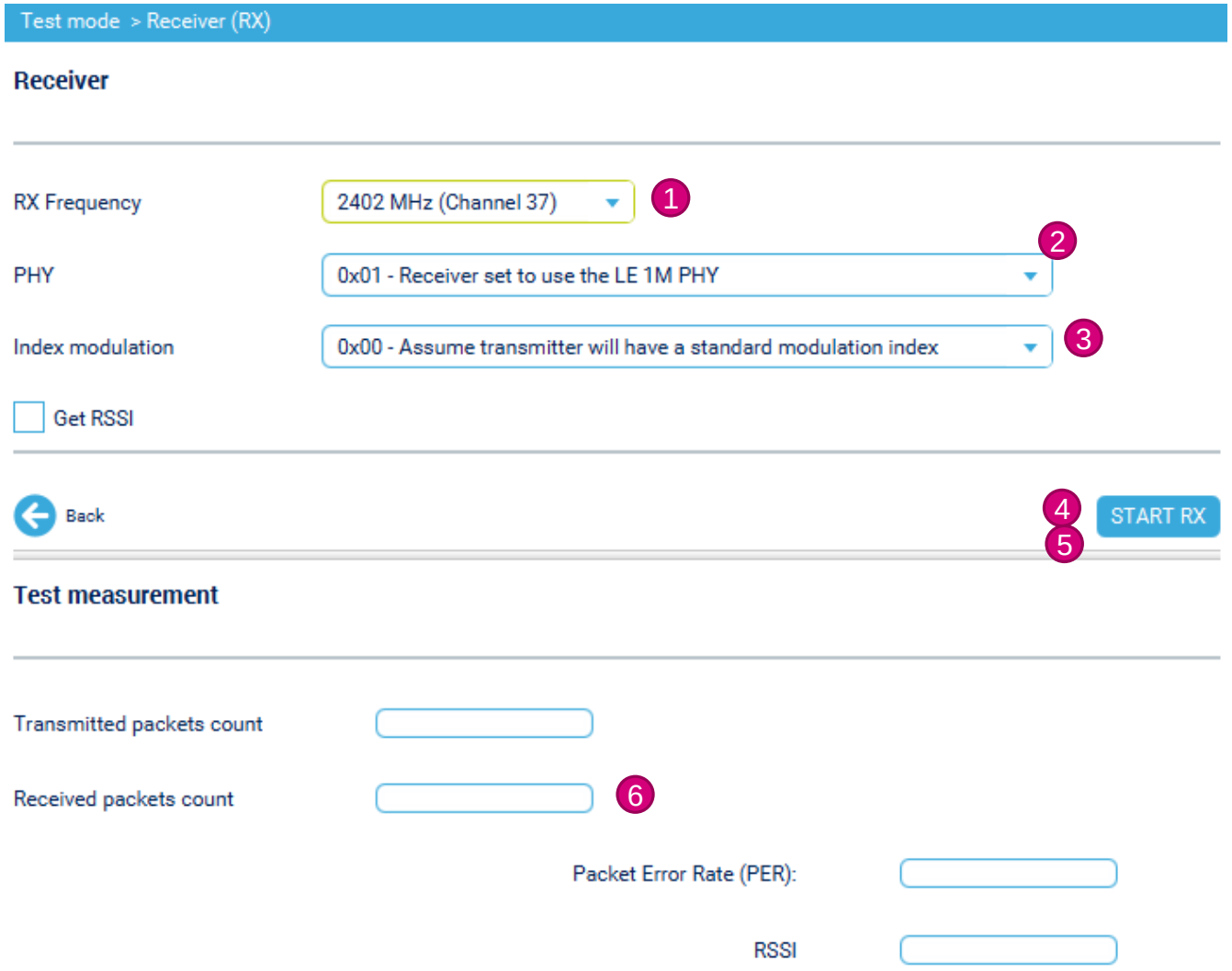
Start/Stop the transmitter test

- To start the packet transmission:
  1. Set the parameters
  2. Click “START TX”
- To stop the packet transmission:
  3. Click “STOP TX”
  4. The transmitted packet count is updated with the number of packets sent by the transmitter



Start/Stop the receiver test

- To start the receiver test:
  1. Set the RX frequency that is a specific BLE RF channel
  2. Set the “PHY” to be used
  3. Set the modulation index
  4. Click “START RX”
- To stop the receiver test:
  5. Click “STOP RX”
  6. The count of received packets is displayed



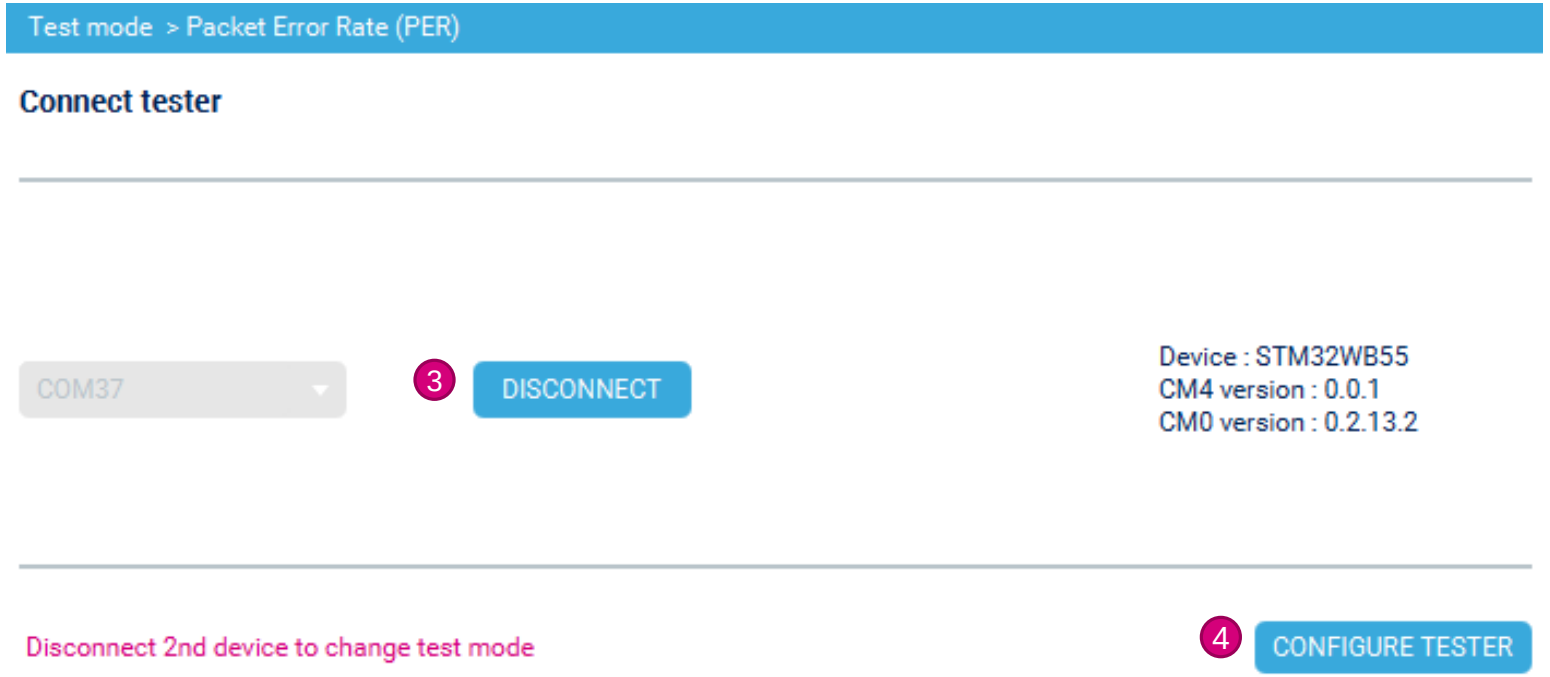
life.augmented

Packet Error Rate (PER)

- Packet error rate is the ratio of packets lost during transmission.
- A device is used as a tester (Tester) to send packets to the Device Under Test (DUT).

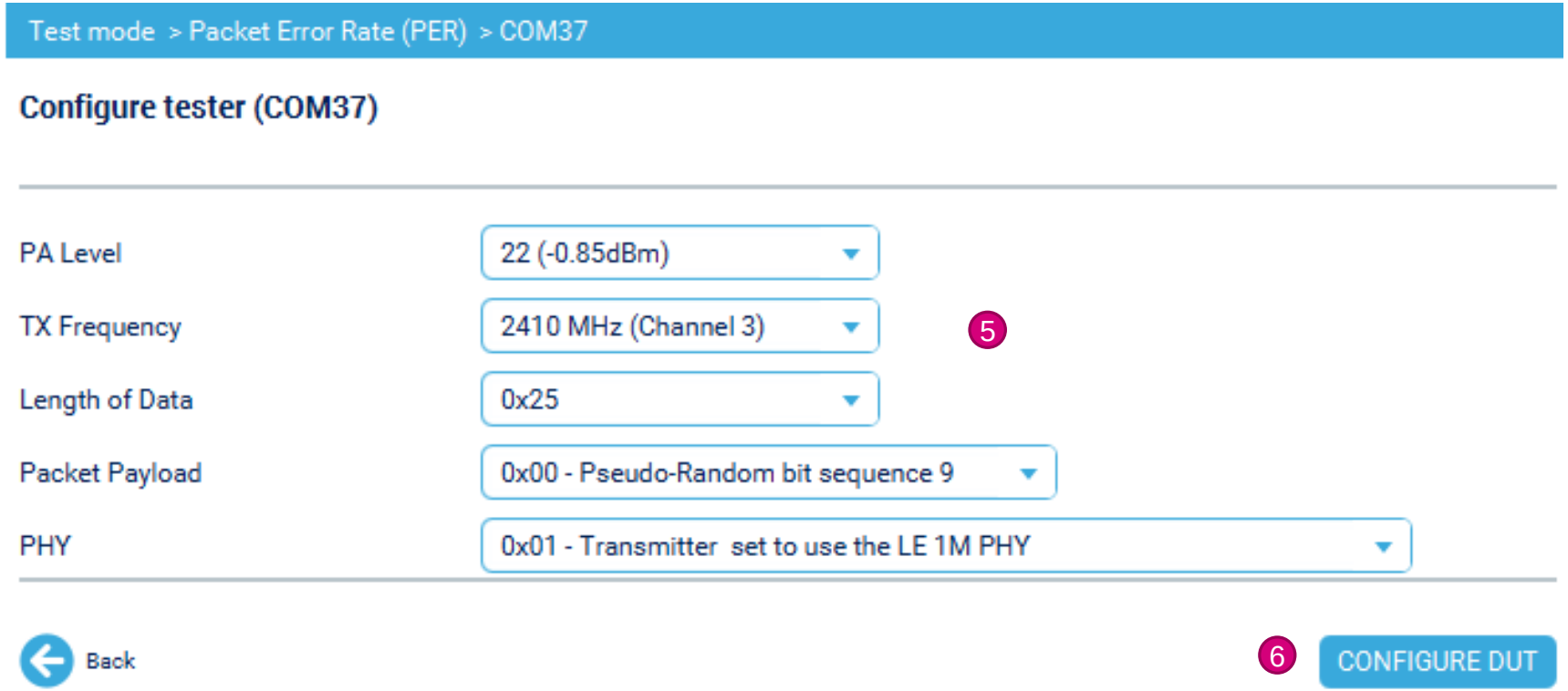
Packet Error Rate test: Step 1

1. Connect the DUT, and open the COM port.
2. In the RF pane, select the PER test mode
3. Connect the “Tester” and open the COM port in the “Connect Tester” area.
4. Click “CONFIGURE TESTER”



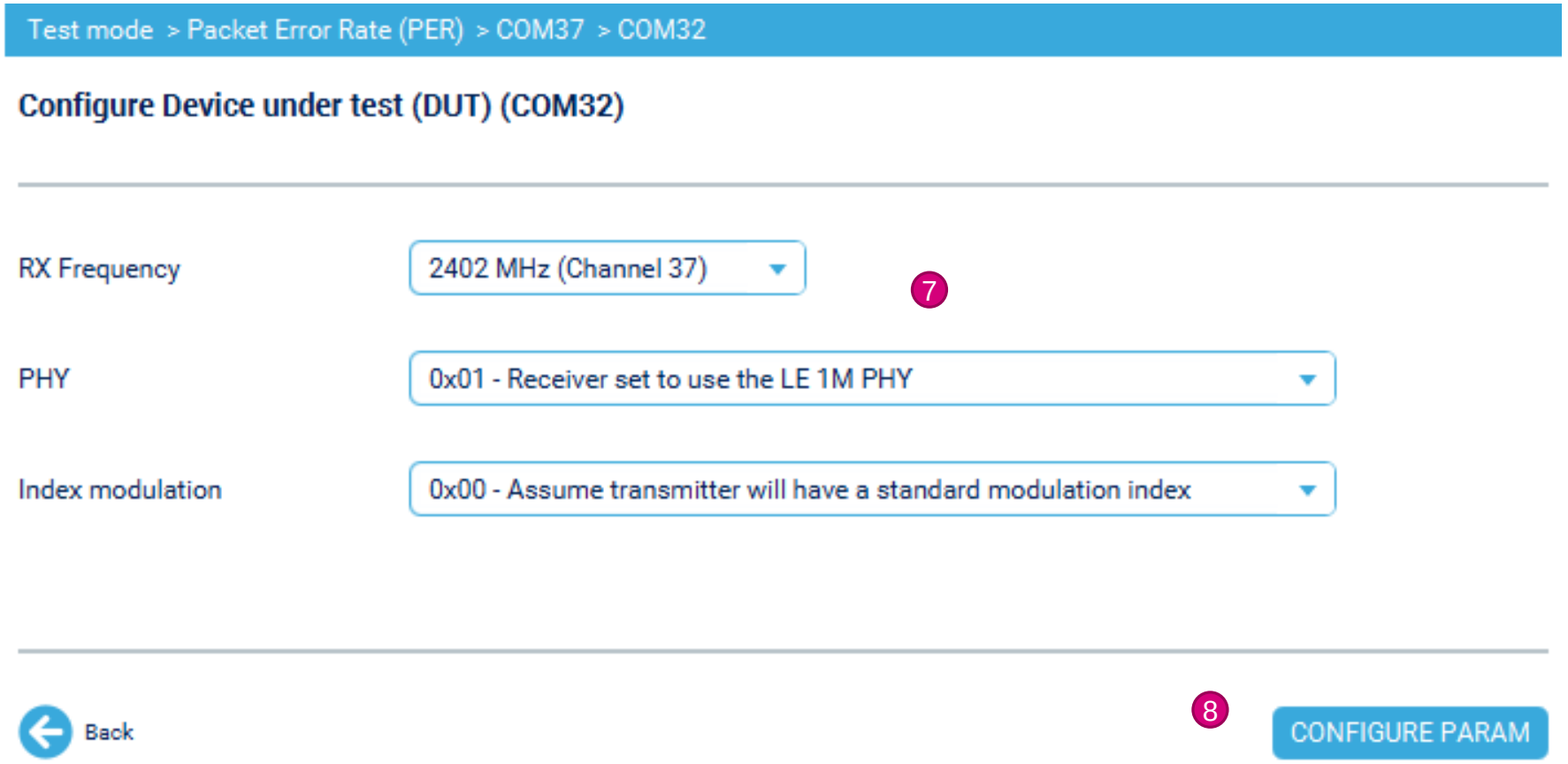
Packet Error Rate test: Step 2

5. Set the tester parameters
6. Click “CONFIGURE DUT”



Packet Error Rate test: Step 3

7. Set the receiver parameters (DUT)
8. Click “CONFIGURE PARAM”

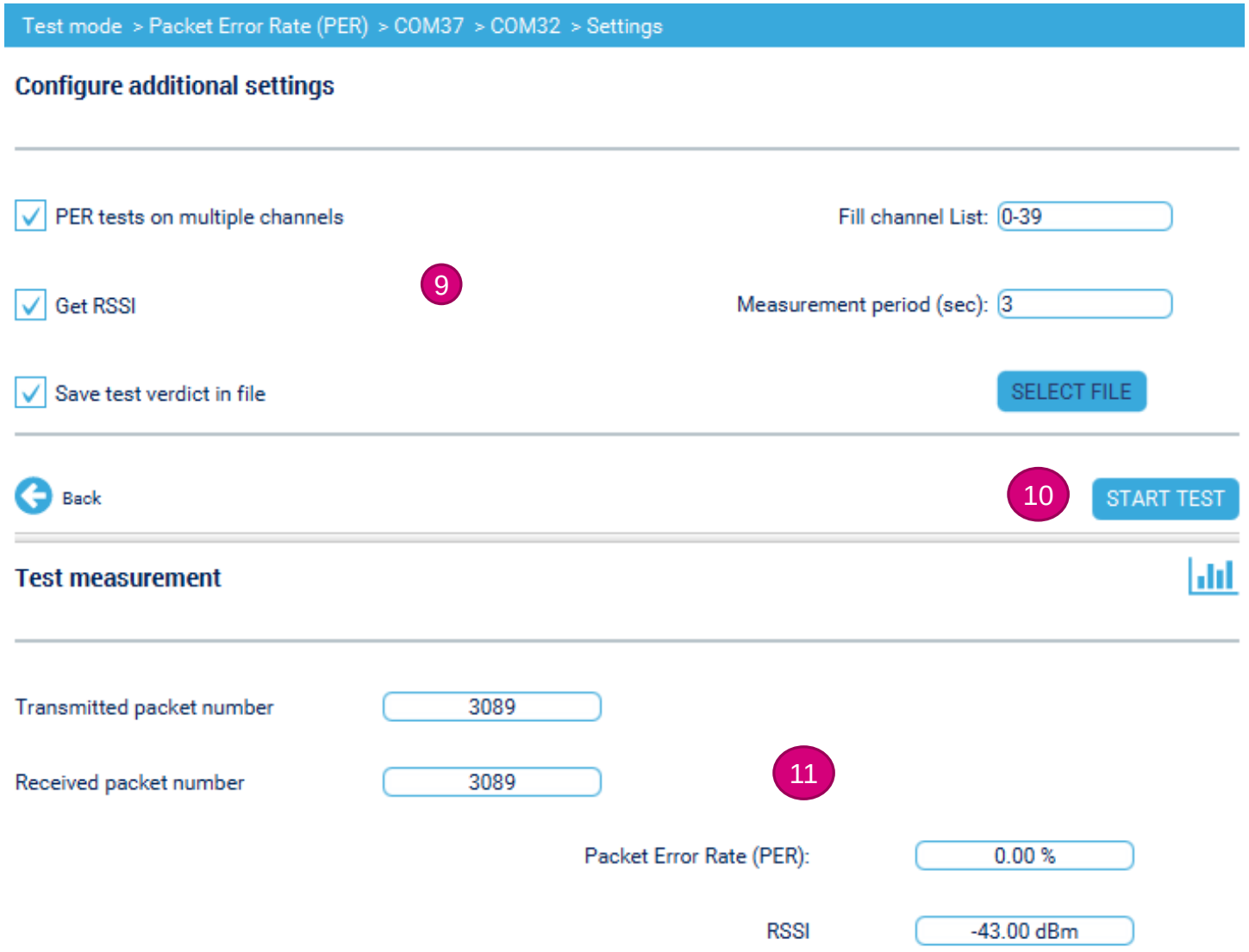


Packet Error Rate test: Step 4

9. Select the test to be performed:
   - Multiple channels
   - With RSSI
   - Save the result in a file.

10. Click “START TEST”

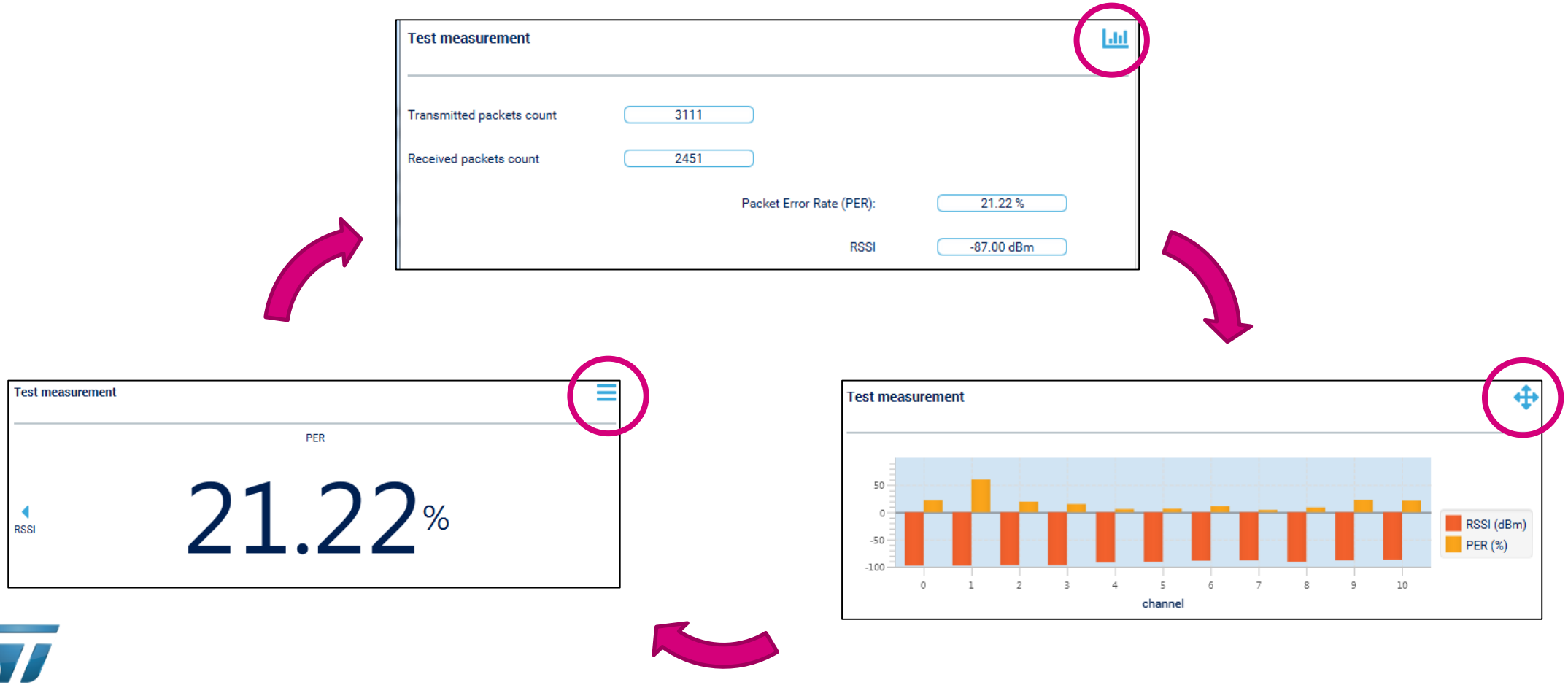
11. The results are displayed in the bottom part when the test is stopped.



life.augmented

Packet Error Rate measurement

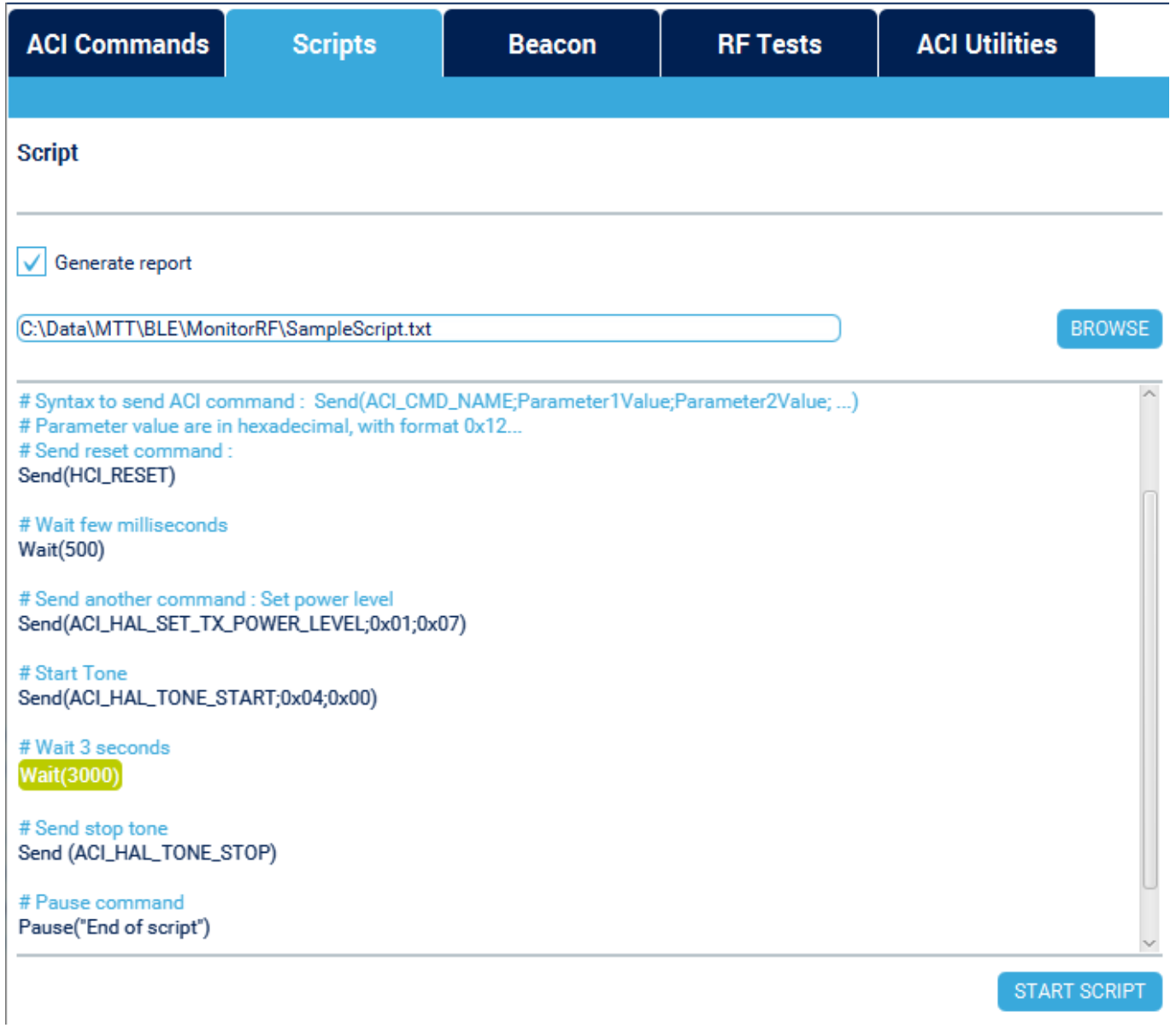
- It is possible to change the display mode between values, graph or large text format using the blue icons at the top right corner .



Scripts

- Scripts are used to send sequences of ACI commands
- Scripts are text files listing the commands to perform.

1. Select the script to load
2. Click “START SCRIPT” to execute the script
3. Use the Start/Stop button to stop or restart the script



life.augmented

Script syntax

Scripts are text files and can be edited

Lines starting with # are comments

To send command, use the “Send” instruction.
The first element is the ACI command name.
The following elements are parameters, in hexadecimal format, starting with 0x.

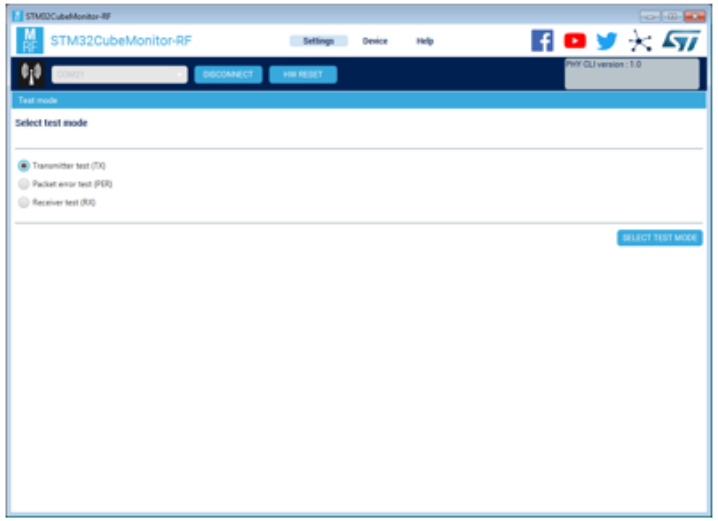
A pause can be added using a “Wait” instruction.
Time is specified in ms.

```
# STM32CubeMonitor-RF sample script
# Line starting with # are comments

# Empty line will be skipped

# Syntax to send ACI command :  Send(ACI_CMD_NAME;Parameter1Value;Parameter2Value; ...)
# Parameter value are in hexadecimal, with format 0x12...
# Send reset command :
Send(HCI_RESET)

# Wait few milliseconds
Wait(500)

# Send another command : Set power level
Send(ACI_HAL_SET_TX_POWER_LEVEL;0x01;0x07)

# Start Tone
Send(ACI_HAL_TONE_START;0x04)

# Wait 3 seconds
Wait(3000)

# Send stop tone
Send (ACI_HAL_TONE_STOP)
```

OpenThread mode
IEEE 802.15.4 RF mode

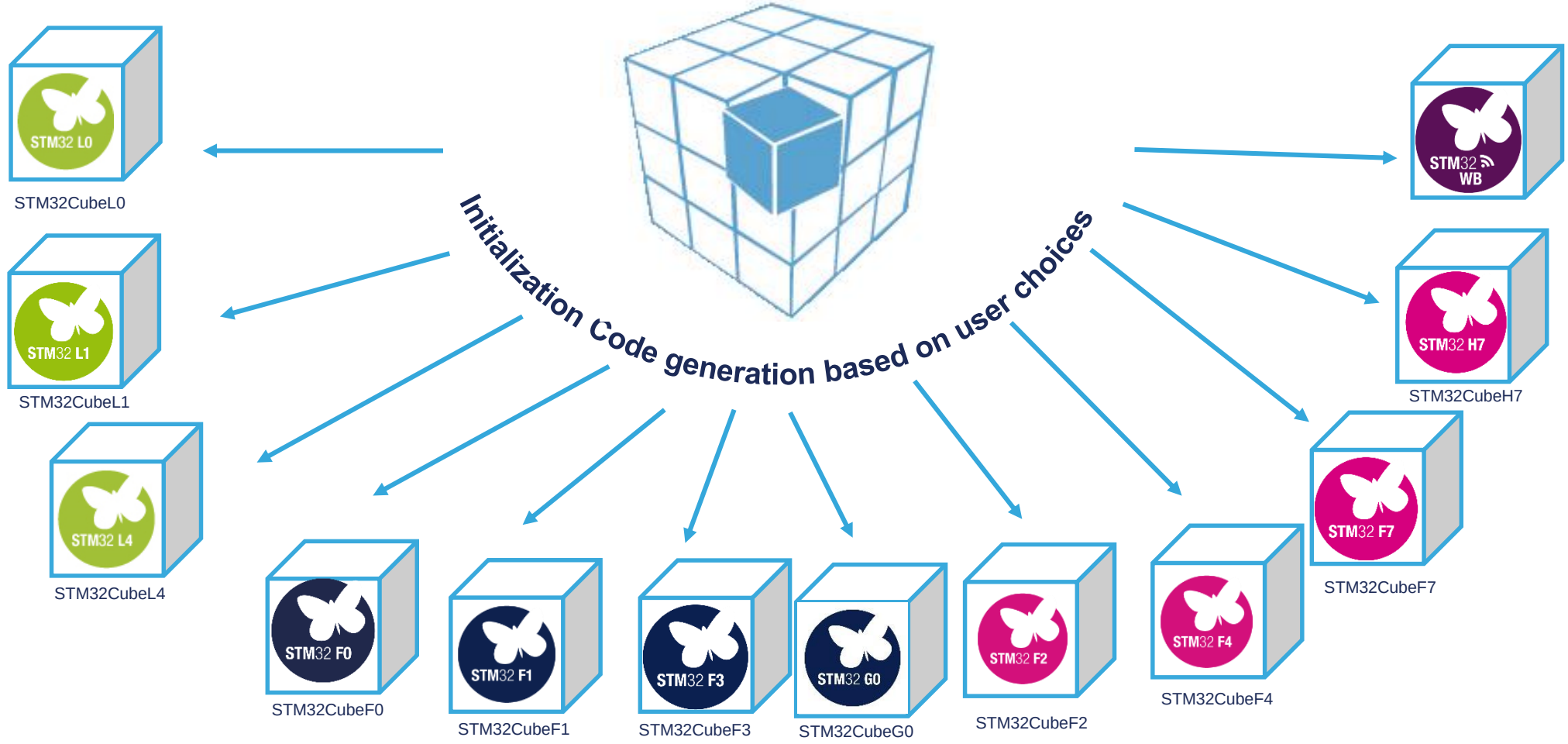
- Thread is an IoT protocol for connected devices in buildings and homes.
- STM32WB devices support the OpenThread stack that implements the Thread protocol OPENTHREAD released by Nest
- The OT (OpenThread) command line API is used to send commands to the stack (CLI).

- IEEE 802.15.4 RF mode is used to test IEEE 802.15.4 devices.
- Thread and Zigbee use IEEE 802.15.4
- The tool allows to:
  - Test transmission,
  - Test reception
  - Measure PER



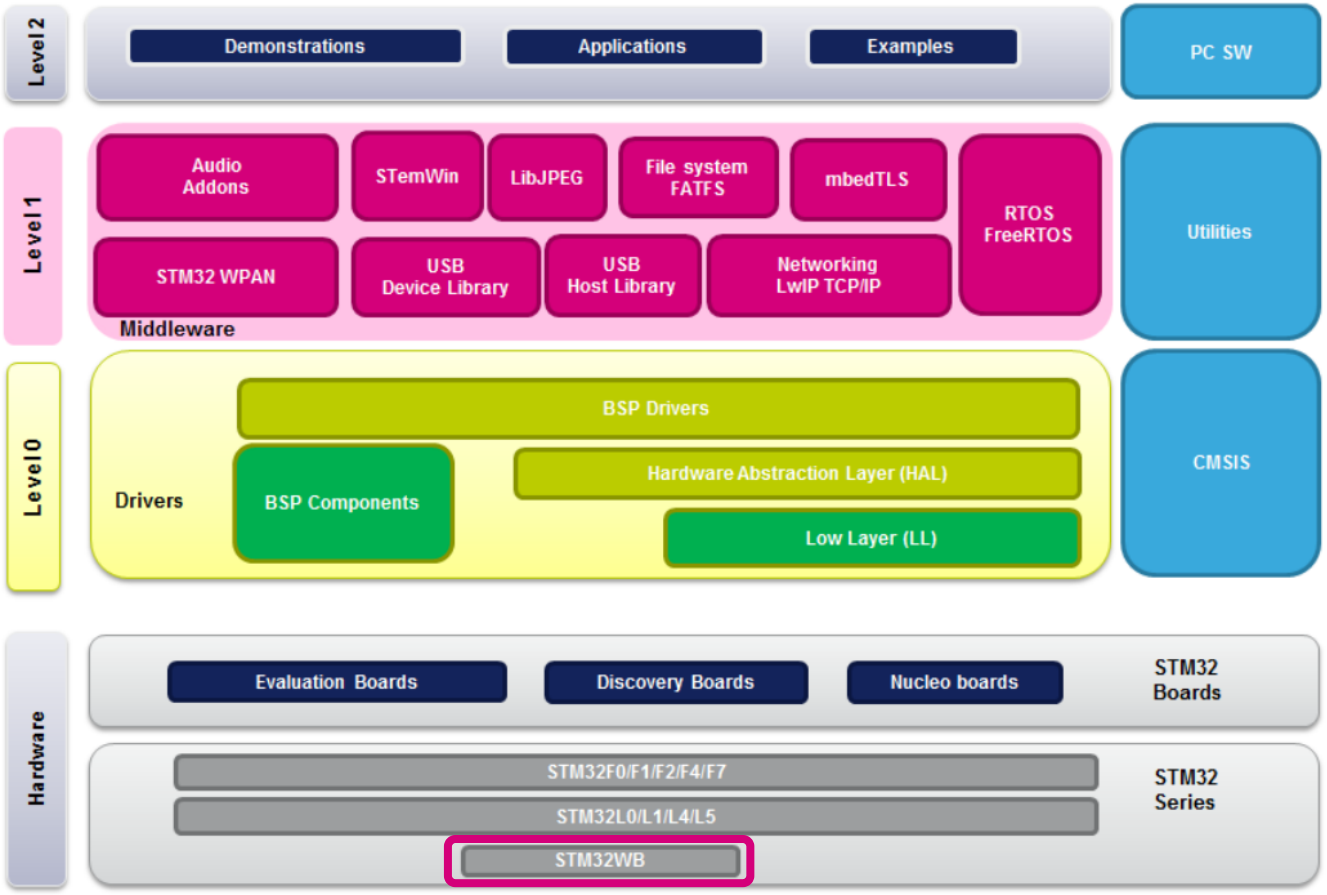
Development environment

STM32CubeMX



life.augmented

Overview



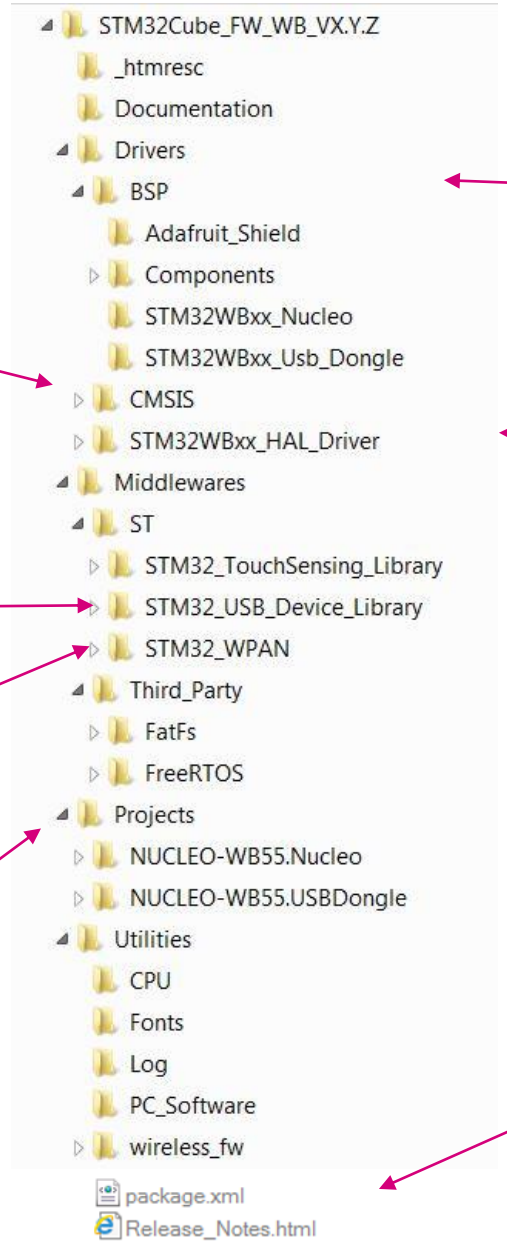
Application benefits

- Single package
- Compatible with all STM32 series
- Source code with open-source BSD license

Key features

Layer	Category	Provided embedded software	Provided examples
HAL	Analog	Analog/Digital conversion, …	~88 examples on STM32WB boards !
	Timers	Timers, RTC, Watchdogs, …	
	Cryptography	CRC, AES, PKA and Random Number generator, …	
	커넥티비티	I2C, USART, SPI, USB, BTH	
	Interface	External memory, LCD, TSC, 시리얼 audio	
Middleware	RTOS	FreeRTOS open source RTOS, with CMSIS-RTOS wrapper	~39 applications on STM32WB boards !
	USB	USB Host and Device cores Device classes: HID, MSC, CDC, Audio, MTP, DFU, and CCID	
	TSC	Touch Sensing controller	
	WPAN	BLE stack, OpenThread stack, BLE & Thread static concurrent mode	
	File System	FatFS open-source file system with enhanced mechanisms including NAND handling	
Application	Demonstration	Full demonstrations for ST boards: BT SIG GATT-based application using ST BlueMS application on IOS/Android smartphone	~22 demonstration projects for ST boards!

Package organization



Contains STM32WBxx CMSIS files that defines Peripheral's registers declarations, bits definition and the address mapping

BSP drivers for the supported boards

STM32WBxx HAL drivers for all modules

Touch Sensing library

USB Device Library supporting both OTG FS and HS cores offering the following classes: HID, MSC, CDC, Audio and DFU

Wireless Personal Area Network library

Set of examples/ applications organized by board provided with preconfigured projects.

Package Release Note providing all the relevant information on its content: main changes, supported devices and boards, supported toolchains, firmware components and versions…

Supported devices & boards

Macro defined in stm32wbxx.h	STM32WB Series devices
STM32WB55xx	STM32WB55VG, STM32WB55CG, STM32WB55RG

Board	Example	Application	Demonstration
STM32WB55 Nucleo pack	88	39	22

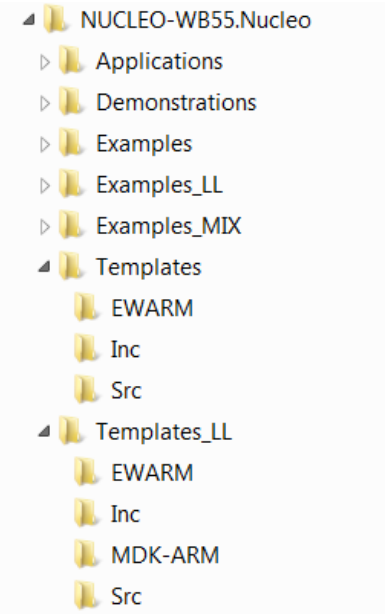
Examples overview

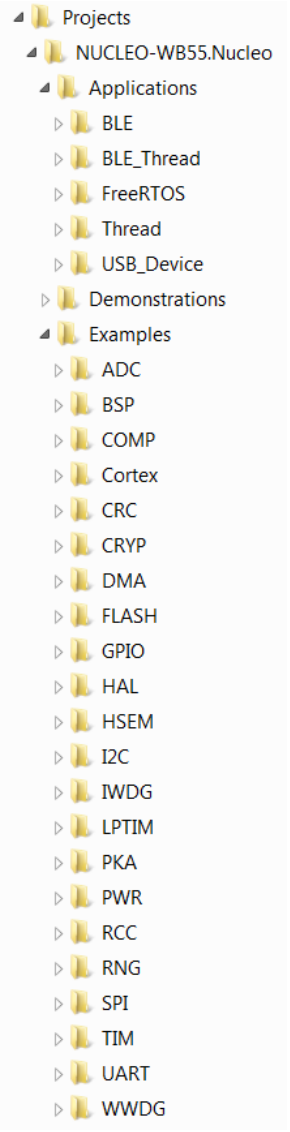
- For each board, a set of examples is provided with preconfigured projects for EWARM, MDK-ARM and SW4STM32 toolchains.
- This figure shows the projects structure for the STM32WB55 NUCLEO board, (order code: P-NUCLEO-WB55) which is identical for other boards.

The examples are classified depending on the STM32Cube level they apply to, and are named as follows:

  - Examples in Level 0 are called **Examples**, and use HAL drivers without any middleware component
  - Examples in Level 1 are called **Applications**, and provide typical use cases of each middleware component
  - Examples in Level 2 are called **Demonstration**, and implement all the HAL, BSP and middleware components

- The **Template** project is provided to build quickly any firmware application for all supported boards
- The **STM32CubeProjectList** file allows quick access and search for a given example within the firmware package
- All examples have the same structure,
  - **\Inc** folder contains all header files
  - **\Src** folder for the source code
  - **\EWARM**, **\MDK-ARM** and **\SW4STM32** contain the preconfigured project for each toolchain.
  - **readme.txt** describes example behavior and the environment needed to make it work.





Documentation

Technical Documentation

Product Specifications			
	Description	Version	Size
	DS11929: Multiprotocol wireless 32-bit MCU Arm®-based Cortex®-M4 with FPU, Bluetooth® 5 and 802.15.4 radio solution	4.0	2 MB
	DB2885: Multiprotocol wireless 32-bit MCU Arm®-based Cortex®-M4 with FPU, Bluetooth® Low Energy and 802.15.4 radio solution	3.1	217 KB

Application Notes			
	Description	Version	Size
	AN4312: Design with surface sensors for touch sensing applications on MCUs	5.0	1 MB
	AN5165: Development of RF hardware using STM32WB microcontrollers	3.0	2 MB
	AN2606: STM32 microcontroller system memory boot mode	36.0	3 MB
	AN5155: STM32Cube MCU Package examples for STM32WB Series	1.0	315 KB
	AN5270: STM32WB Bluetooth™ low energy (BLE) wireless interface	1.0	1 MB
	AN5071: STM32WB ultra-low-power features overview	2.0	459 KB
	AN2834: How to get the best ADC accuracy in STM32 microcontrollers	3.1	1 MB
	AN4229: How to implement a vocoder solution using STM32 microcontrollers	1.1	466 KB

Programming Manuals			
	Description	Version	Size
	PM0214: STM32 Cortex®-M4 MCUs and MPUs programming manual	7.0	3 MB

- Full set of documentation available ranging from datasheet and register manual to application and technical notes.

STM32 Generic doc

STM32 WB Series specific doc
its content is similar to the **STM32CubeProjectList** file available within the STM32CubeWB firmware Package

STM32 WB Series specific doc

STM32 WB Series specific doc

STM32 Generic doc

STM32 Generic doc

STM32 Generic doc

Reference

Exhaustive documentation list and STM32CubeWB Firmware package can be accessed from ST's web site at www.st.com/stm32cubefw

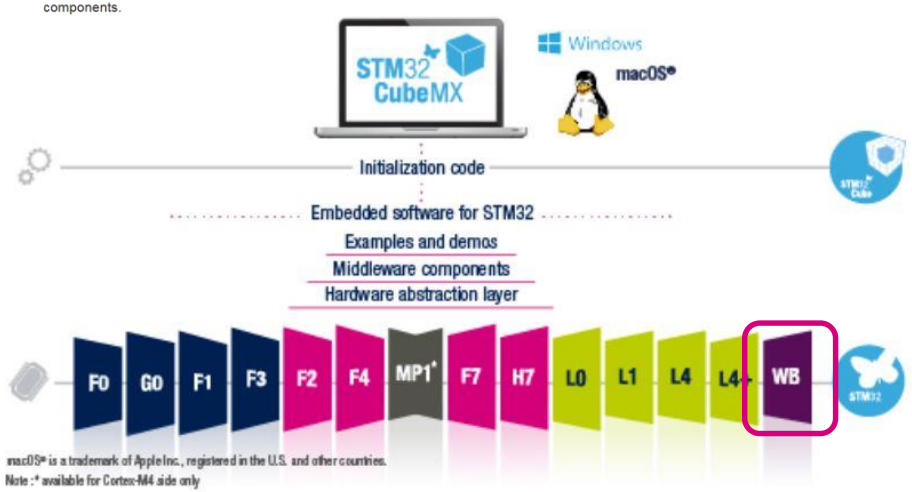
STM32Cube MCU & MPU Packages

STM32Cube is a set of tools and embedded software bricks available free of charge to enable fast and easy development on the STM32 platform which simplifies and speeds up developers' work.

A large number of code use examples are also included making it even easier to get started.

STM32Cube consists of the following components that can be used together or independently:

- The STM32CubeMX graphical user interface and initialization code generator that:
  - Provides graphical wizards to generate initialization C code and includes a utility tool for assisting developers with pin multiplexing, clock tree setting, peripheral configurations and setting up the middleware
  - Generates IDE-ready projects for a wide selection of integrated development environment toolchains
  - Calculates the power consumption for user-defined application sequences
  - Directly imports STM32Cube embedded software libraries from st.com
  - Keeps STM32CubeMX software up-to-date thanks to an Integrated updater
- STM32Cube MCU and MPU Packages for each individual STM32 MCU and MPUs series that include:
  - The hardware abstraction layer (HAL) enabling portability between different STM32 devices via standardized API calls
  - Low-layer (LL) APIs, a light-weight, optimized, expert oriented set of APIs designed for both performance and runtime efficiency
  - A collection of middleware components including RTOS, USB library, file system, TCP/IP stack, touch-sensing library or graphics library (depending on the STM32 series)
  - For STM32 MPUs only, the BSP drivers are based on HAL drivers and provide an API Set to the evaluation board and 3rd party components.

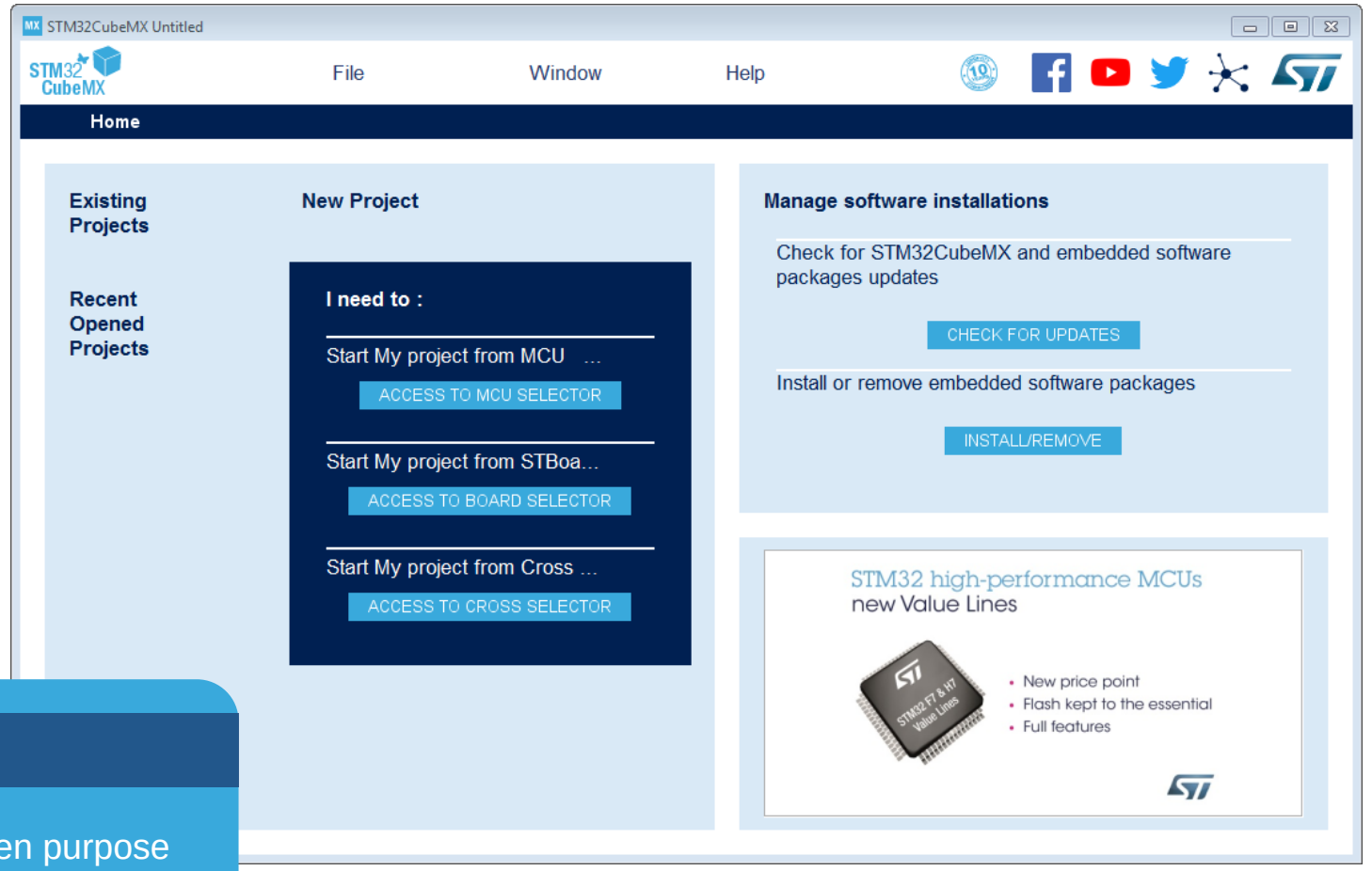


Part Number	Supplier	Supported Devices	STM32CubeMX Compatible
Part Filter			
STM32CubeH7 STM32Cube MCU Package for STM32H7 series (HAL low level drivers, USB, TCP/IP, File system, RTOS,...	ST	STM32H7	true
STM32CubeMP1 STM32CubeMP1 Package for STM32MP1 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32MP1	true
STM32CubeWB STM32Cube MCU Package for STM32WB series (HAL and LL low level drivers, USB, File system, RTOS -...	ST	STM32WB	true
STM32CubeF0 STM32Cube MCU Package for STM32F0 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32F0	true
STM32CubeF1 STM32Cube MCU Package for STM32 F1 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32F1	true
STM32CubeF2 STM32Cube MCU Package for STM32 F2 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32F2	true
STM32CubeF3 STM32Cube MCU Package for STM32 F3 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32F3	true
STM32CubeF4 STM32Cube MCU Package for STM32F4 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32F4	true
STM32CubeF7 STM32Cube MCU Package for STM32F7 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32F7	true
STM32CubeG0 STM32Cube MCU Package for STM32G0 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS),...	ST	STM32G0	true

life.augmented

STM32CubeMX

- Choose ideal MCU and simply configure
  - Pinouts
  - Clocks and oscillators
  - Peripherals
  - Low-power modes
  - Middleware



Application benefits

- Helps choose the correct MCU for a given purpose
- Simulation provides an advantage in design phase
- Boosts development speed with a headstart

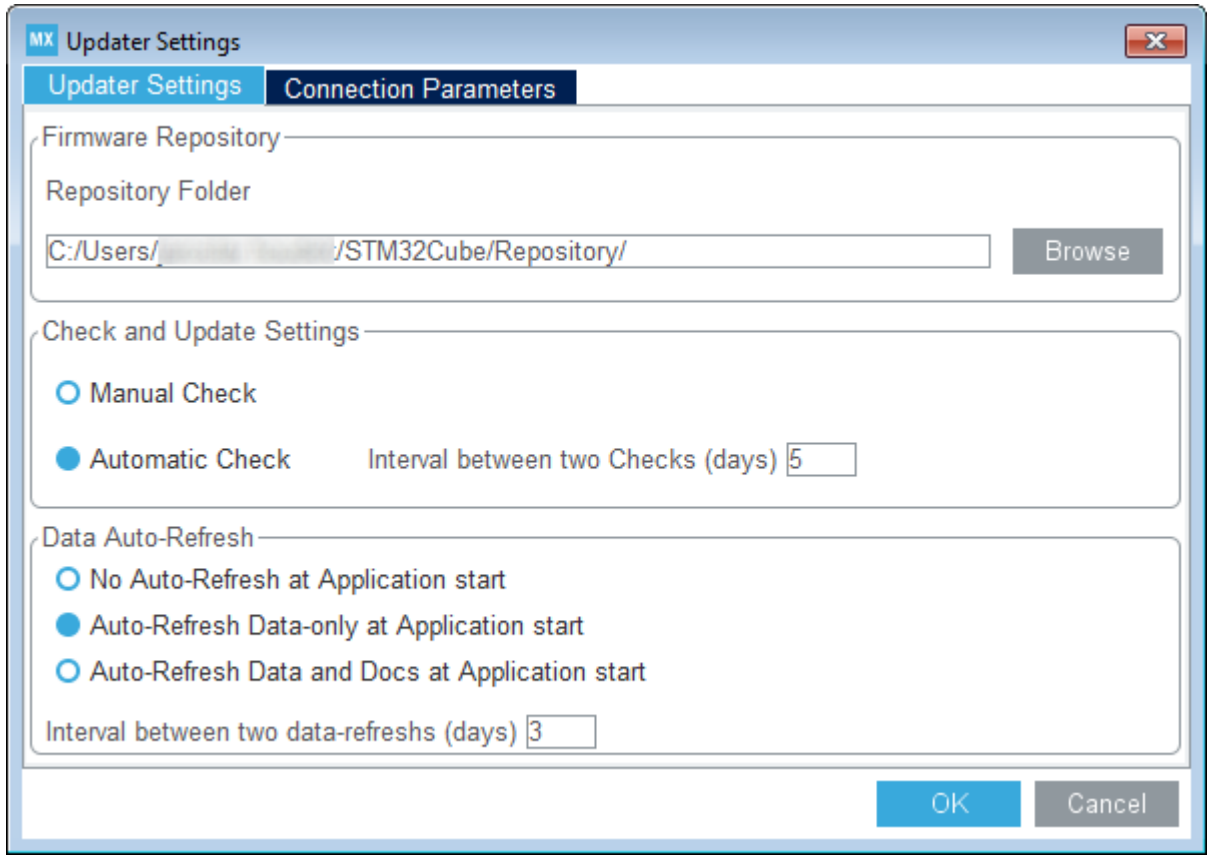
life.augmented

Key features

- Peripheral and middleware parameters
- Power consumption calculator
- Code generation
  - Possible to re-generate code while keeping user code intact.
- Option of command-line and batch operation
- Expandable by plugins
- MCU selector
  - Filter by family, package, peripherals or memory sizes.
  - Search for similar product.
- Pinout configuration
  - Choose peripherals to use and assign GPIO and alternate functions to pins.
- Configure NVIC and DMA
- Clock tree initialization
  - Choose oscillator and set PLL and clock dividers.

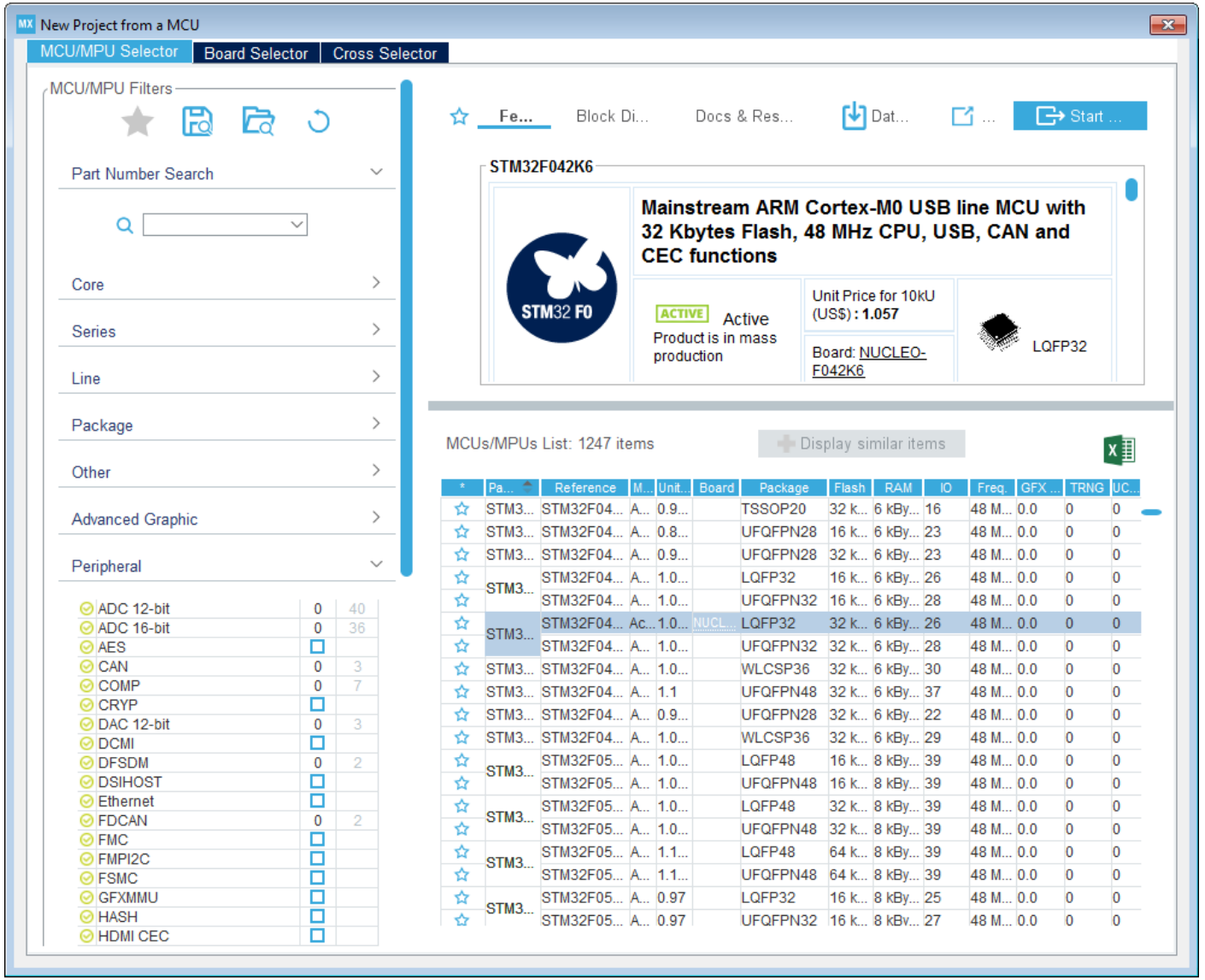
Prerequisites and settings

- STM32CubeMX needs Java RE
  - Check release notes of the particular version for additional requirements.
  - Multiplatform tool runs on Windows, Linux and macOS
- After installation, hit Alt+S to configure the updater – not only for the GUI but also for Cube FW libraries.
- Select SW library placement.



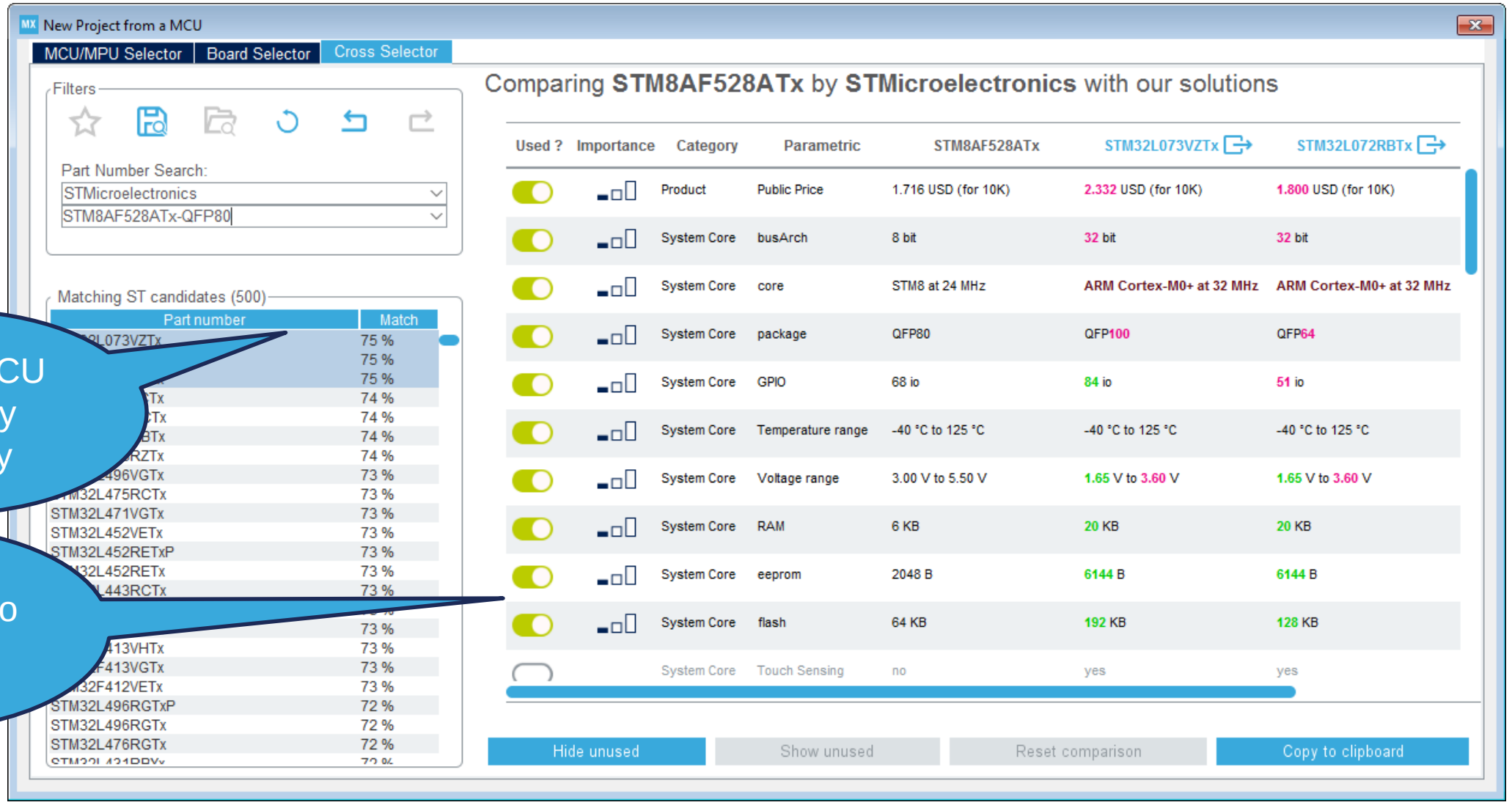
MCU selector

- Find MCU by name …
  - Quickly locate by Series and Lines
- … or application needs
  - Package (pin count)
  - RAM size
  - NV memory requirements
  - Embedded peripherals
  - Number and type of interfaces
  - Core and frequency
  - Price
- Convenient links to documentation
- Export table to Excel file



life.augmented

MCU Cross Selector



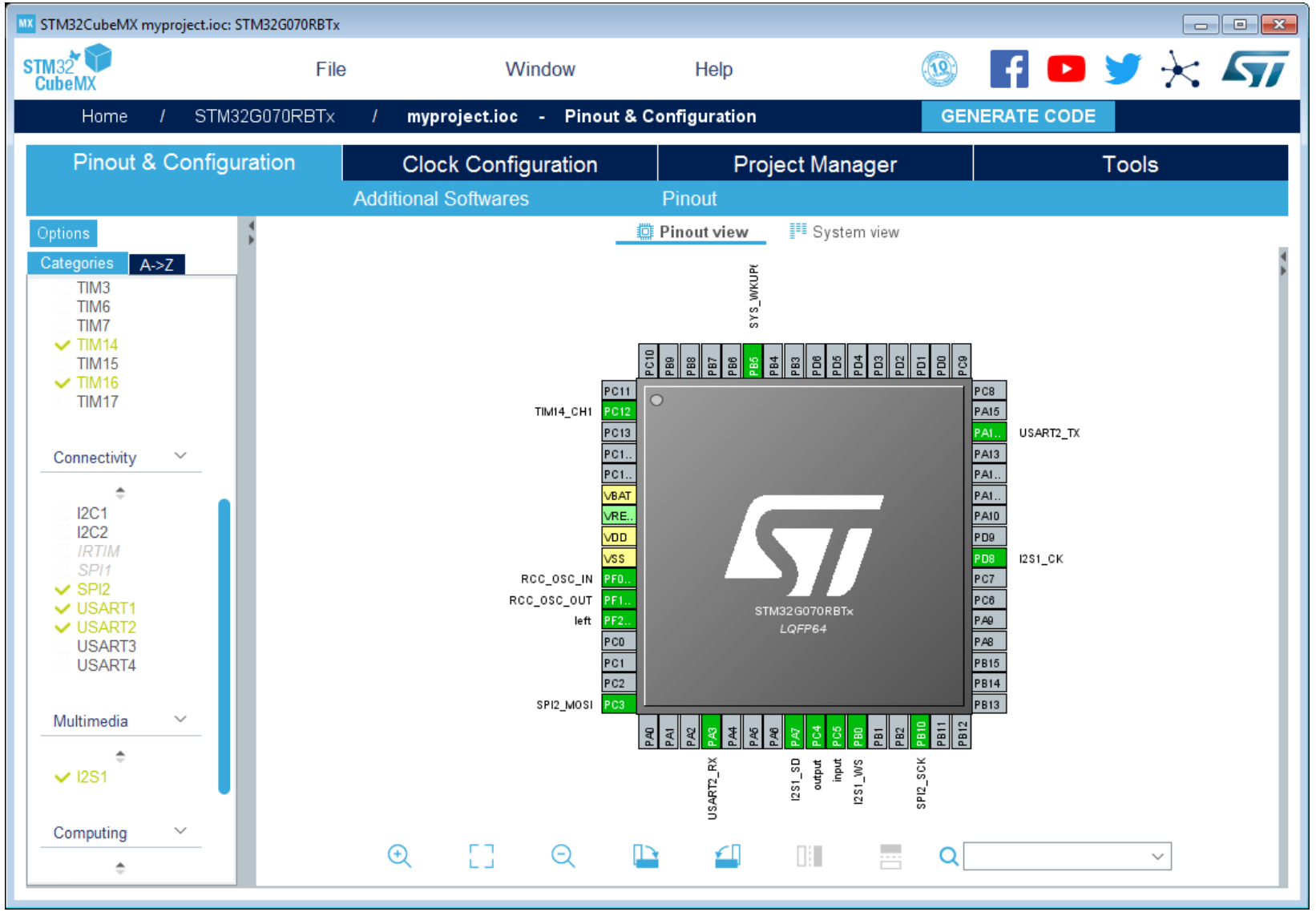
STM32 MCU sorted by similarity

Detailed comparison to selected product

life.augmented

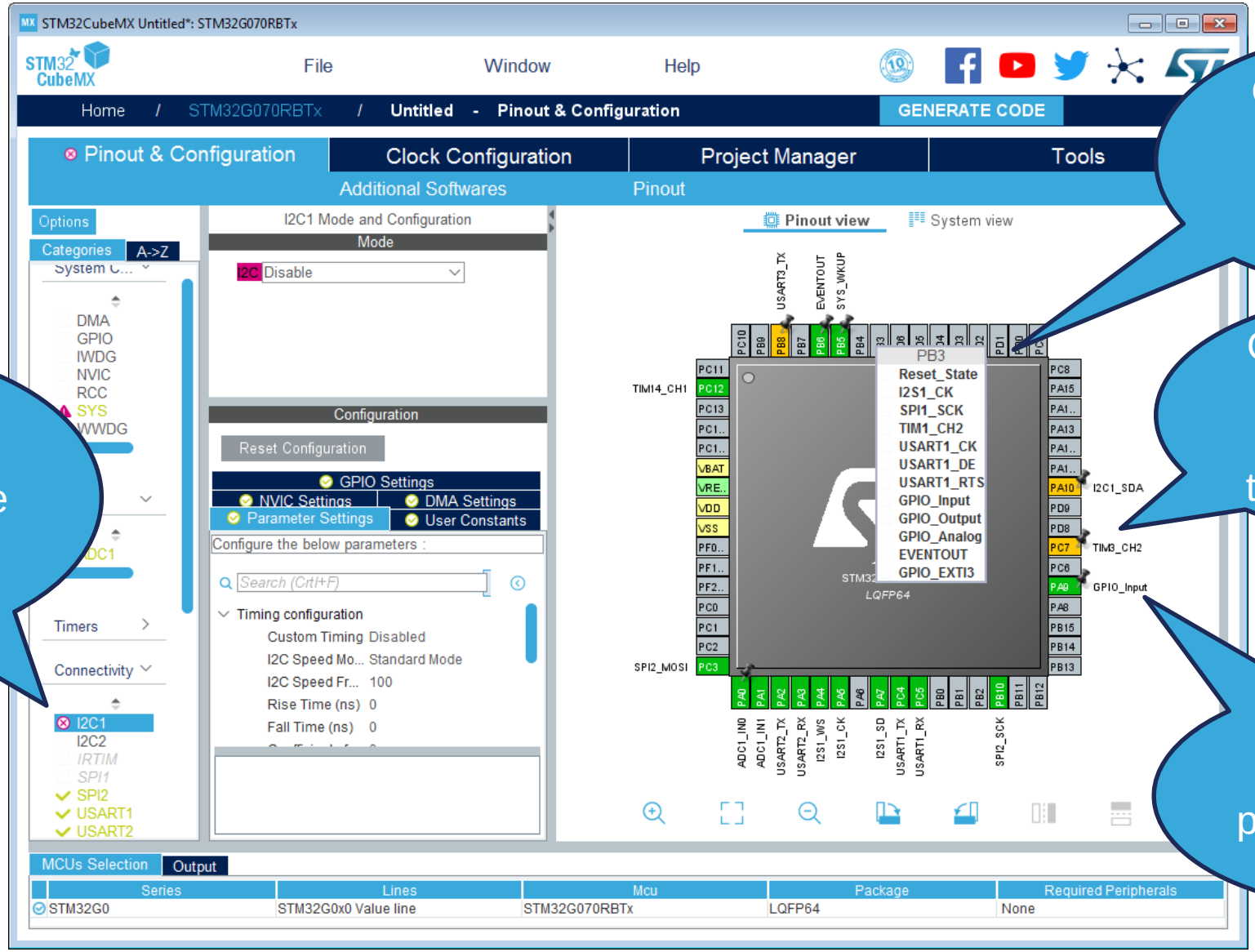
Pin assignment

- Pinout from:
  - Peripheral tree
  - Manually
- Automatic signal remapping
- Management of dependencies between peripherals and/or middleware (FatFS, USB …)



life.augmented

Pin assignment (cont.)



Peripheral is not available, all its alternate pins are assigned elsewhere.

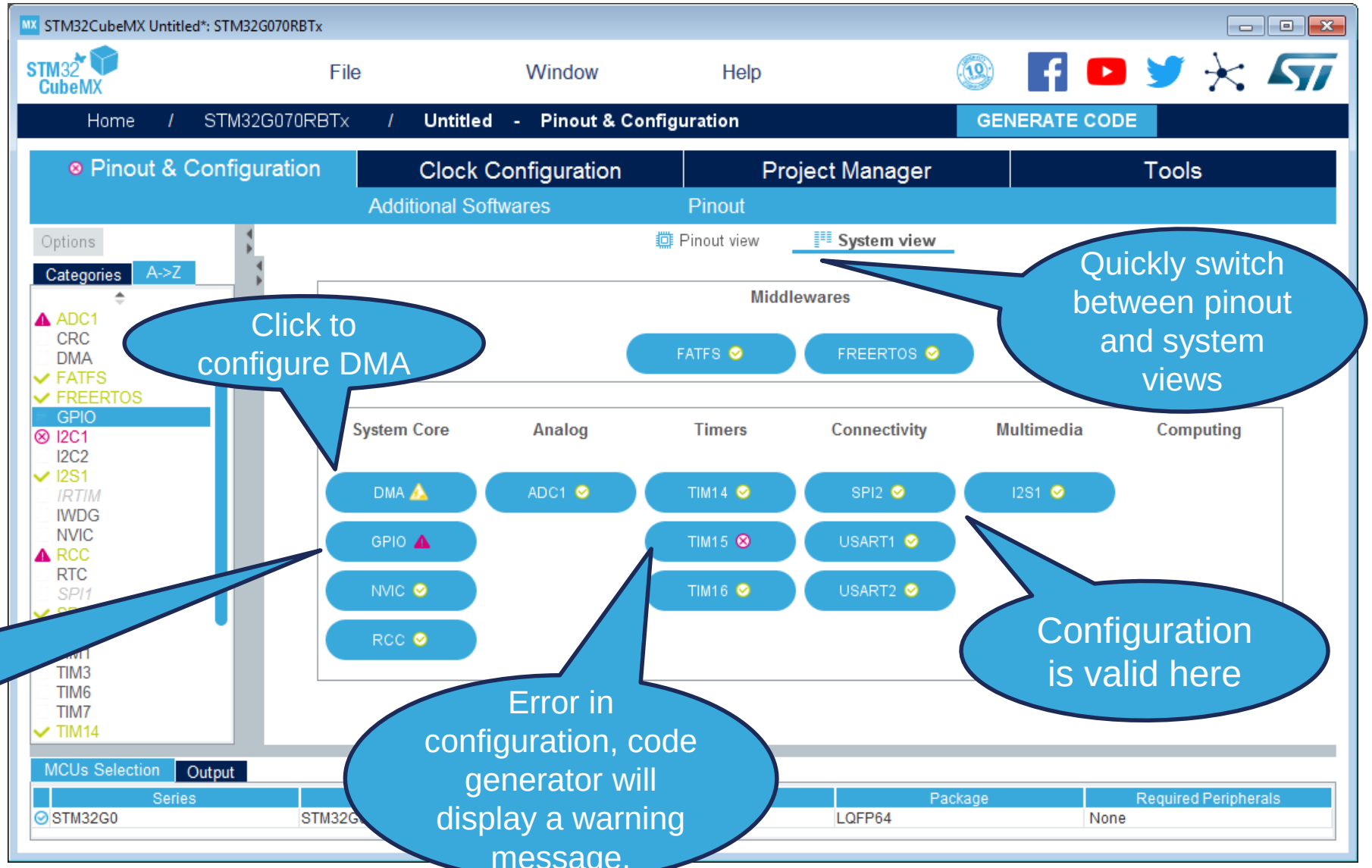
Click on the pin to view alternate functions

Orange warns that the peripheral is not enabled, only the pin is assigned

Freeze the signal placement using the pin icon

Peripheral and Middleware configuration

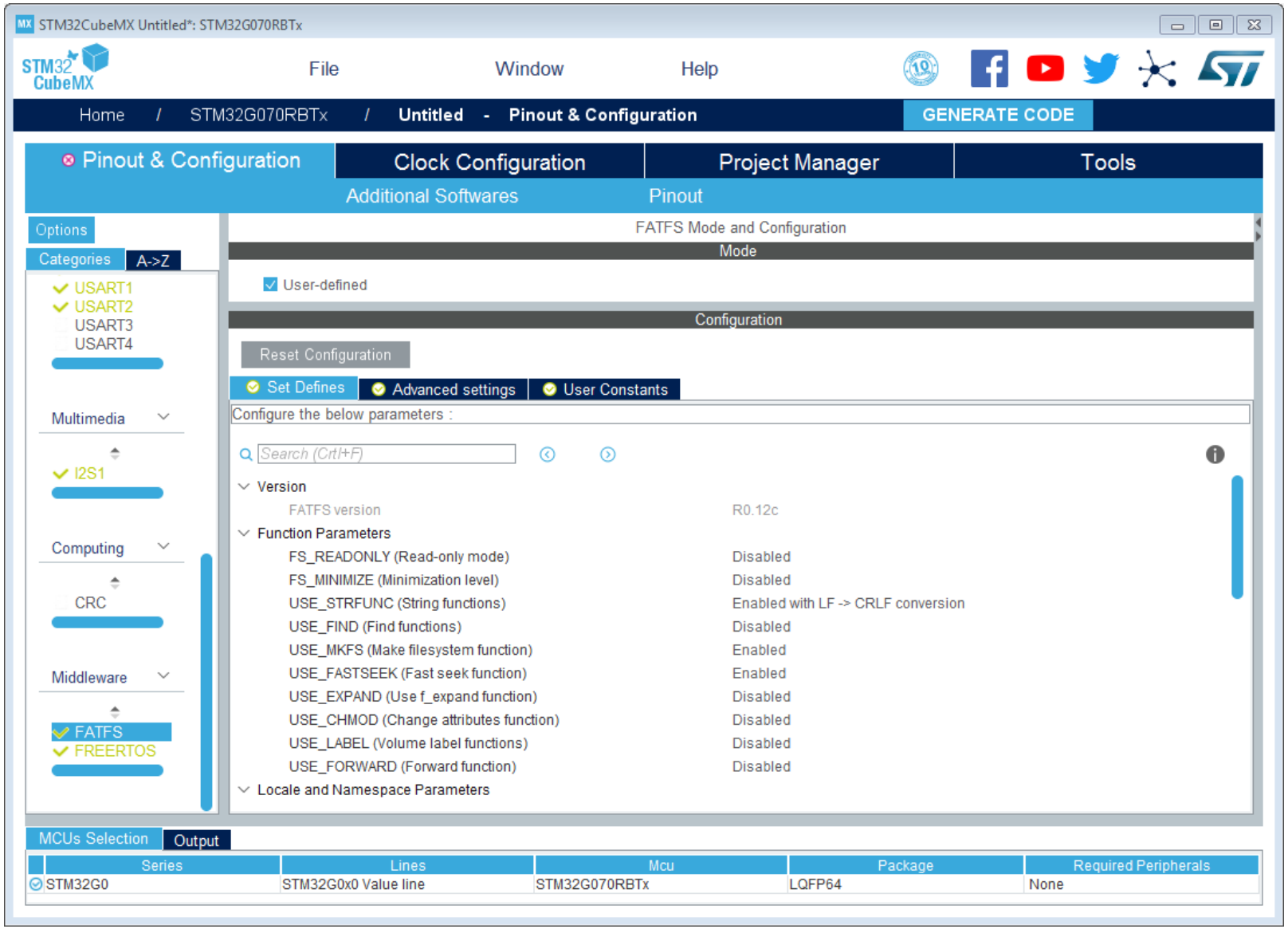
- Global view of used peripherals and middleware.
- Highlight of configuration errors
  - + Not configured
  - v OK
  - ⚠ Non-blocking problem
  - x Error



life.augmented

Middleware configuration

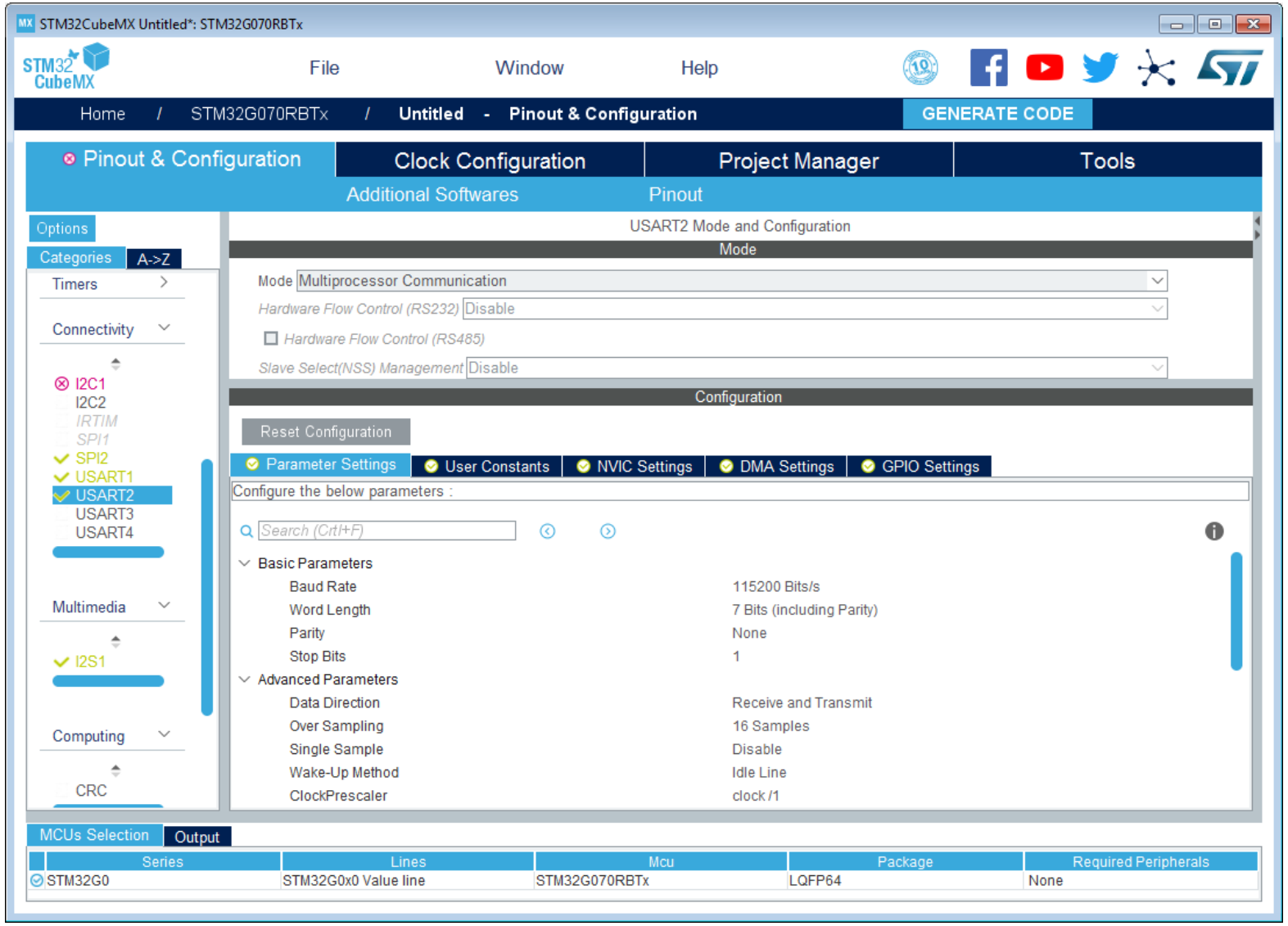
- Presents options specific to each supported software component.
- All settings are organized in logical groups.
- Description and constraints are available for quick reference.



life.augmented

Peripheral configuration

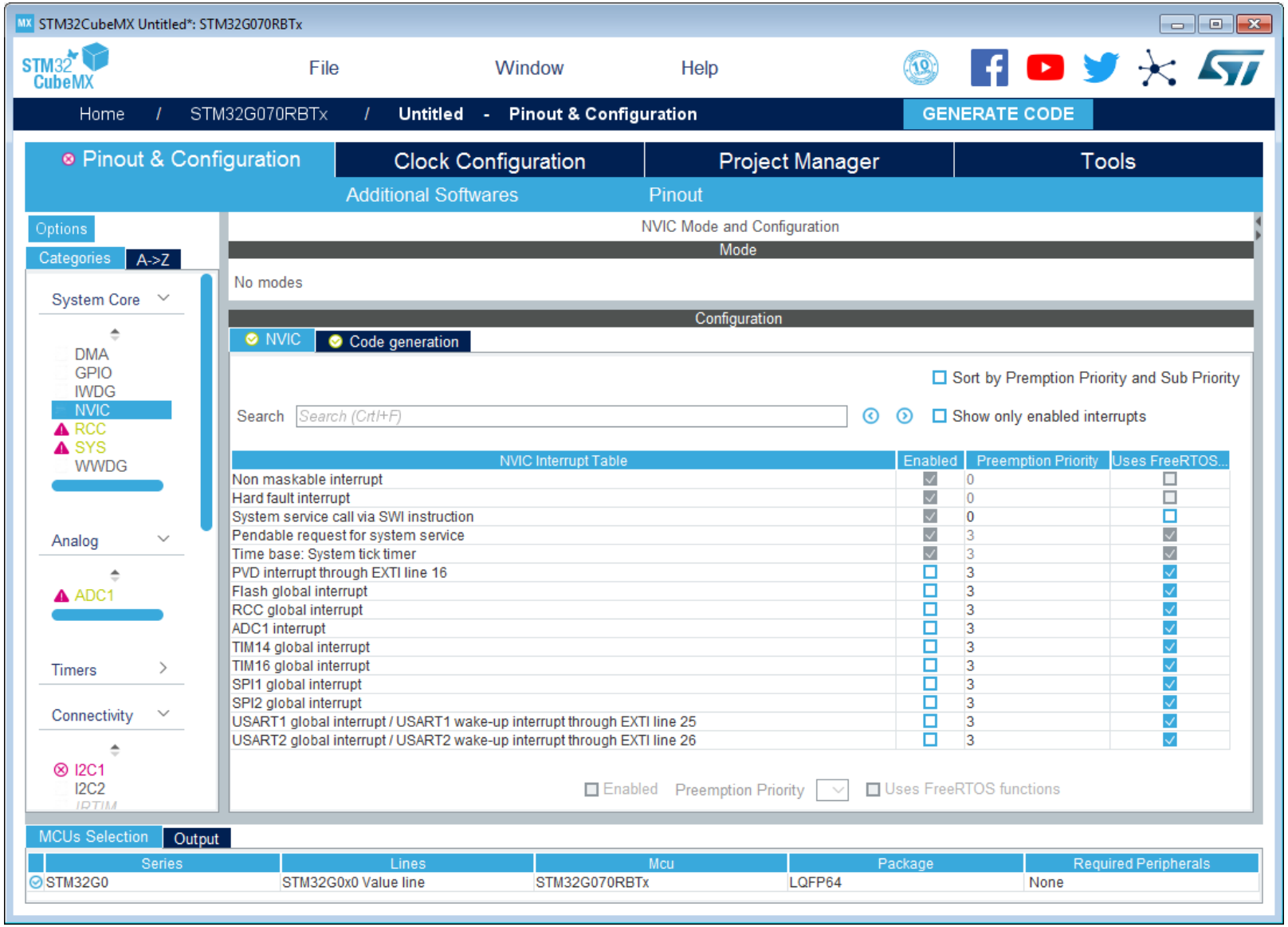
- All available initialization parameters are presented with short description and options.
- Interrupt may be assigned to peripherals.
- DMA may be associated, where applicable.
- GPIO settings for peripherals with input and/or output.



life.augmented

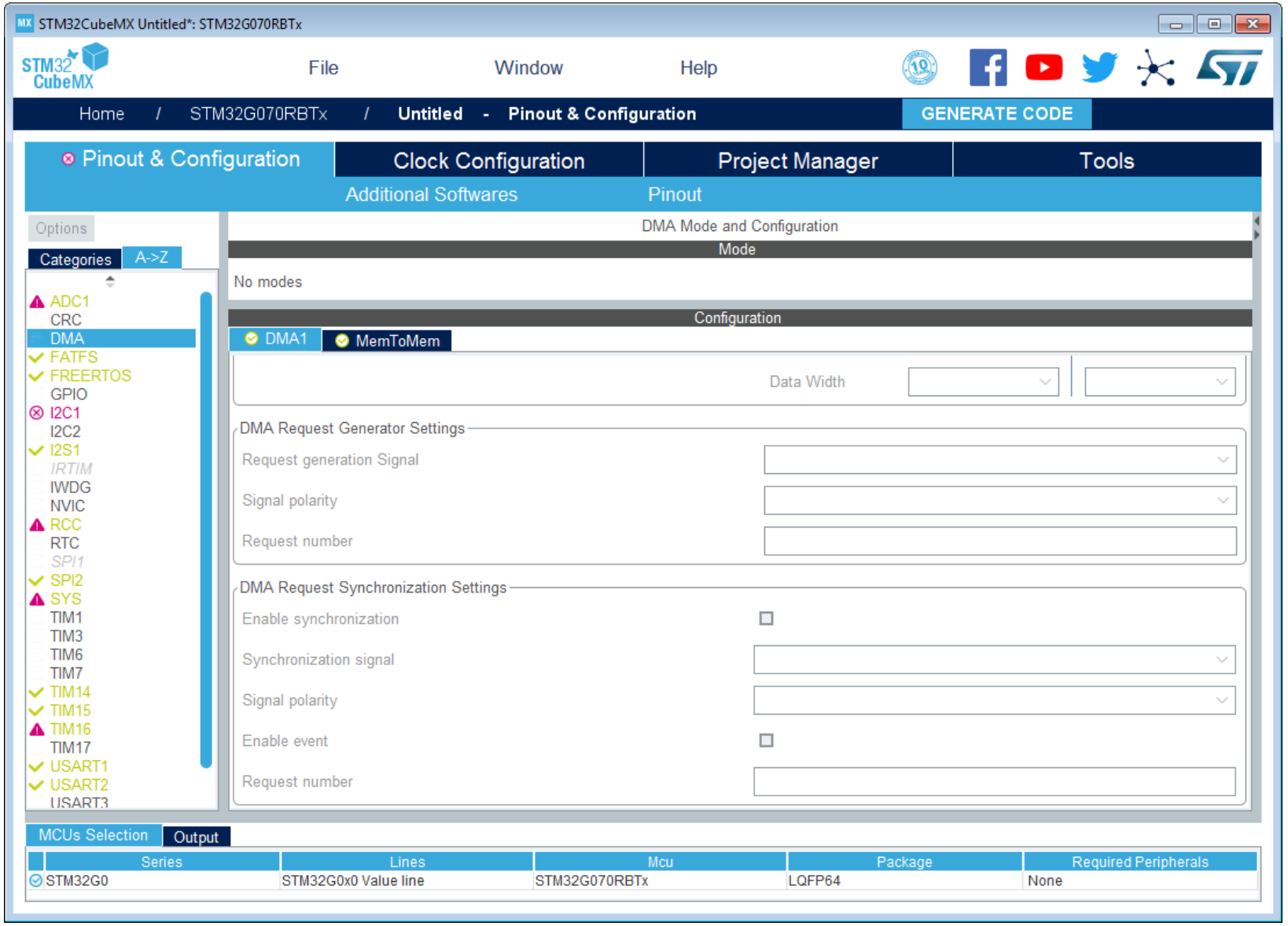
NVIC configuration panel

- Single control panel for all interrupts.
- Manage priorities and sub-priorities.
- Searching, filtering and sorting interrupts in the list.
- Code generation tab allows to customize interrupt initialization.



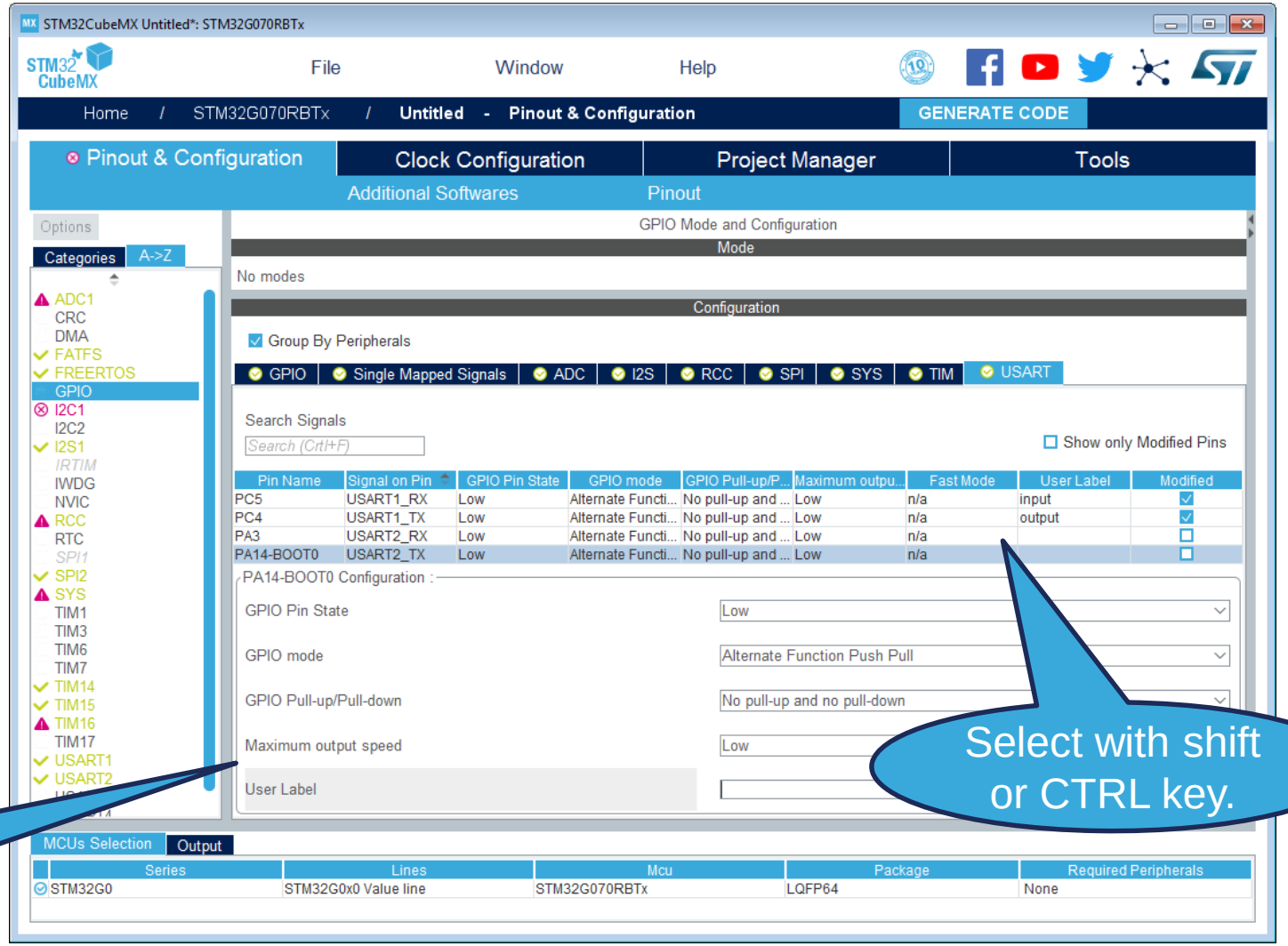
DMA configuration panel

- Manages all DMA requests including memory to memory.
- Configure direction, priority and other settings.



GPIO settings panel

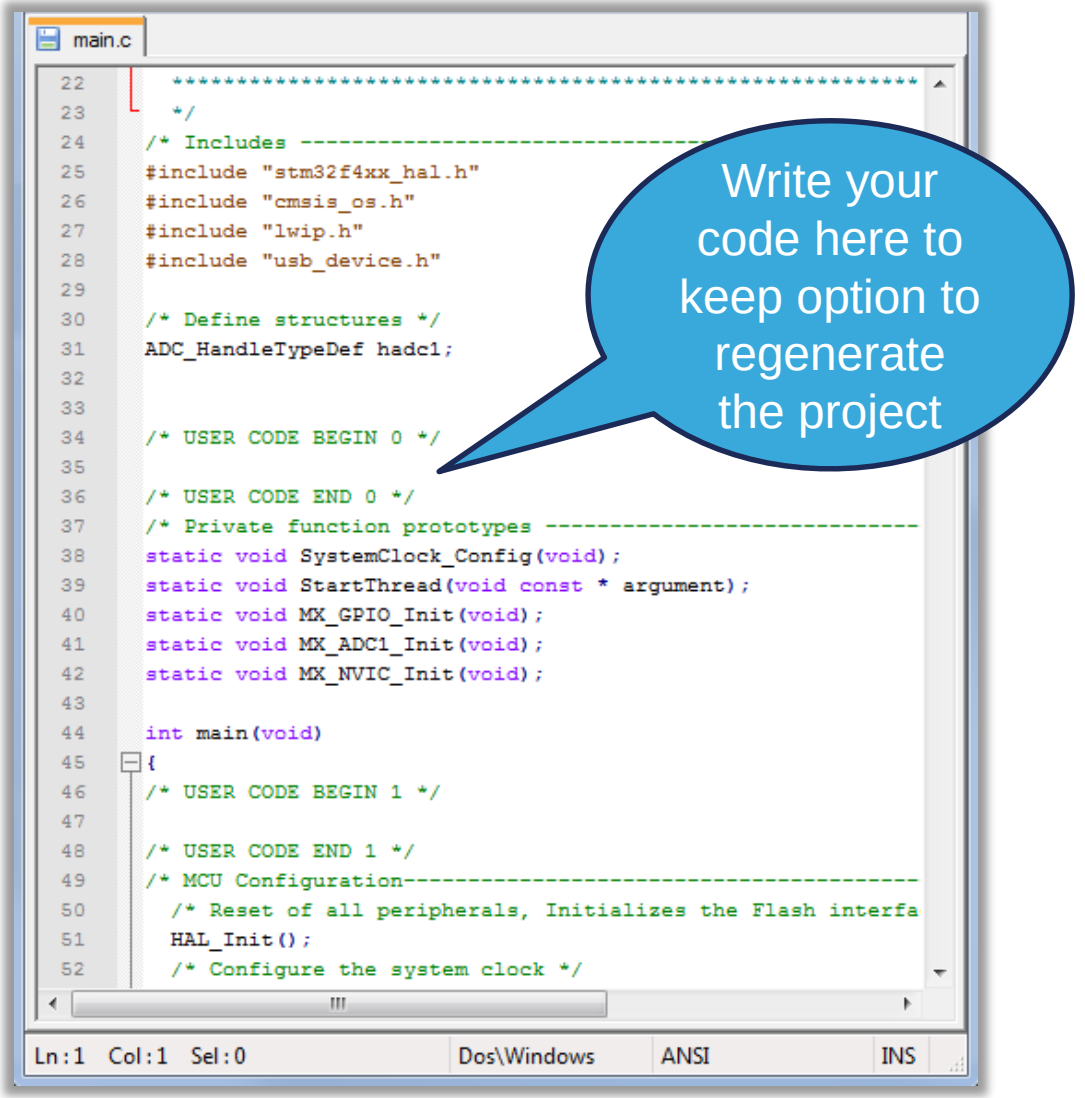
- The application attempts to set sensible default values for most GPIO parameters.
- Default values are chosen, as low-speed and no pull-up.
- Multiple pins can be selected to set them to the same configuration.



life.augmented

Code generation

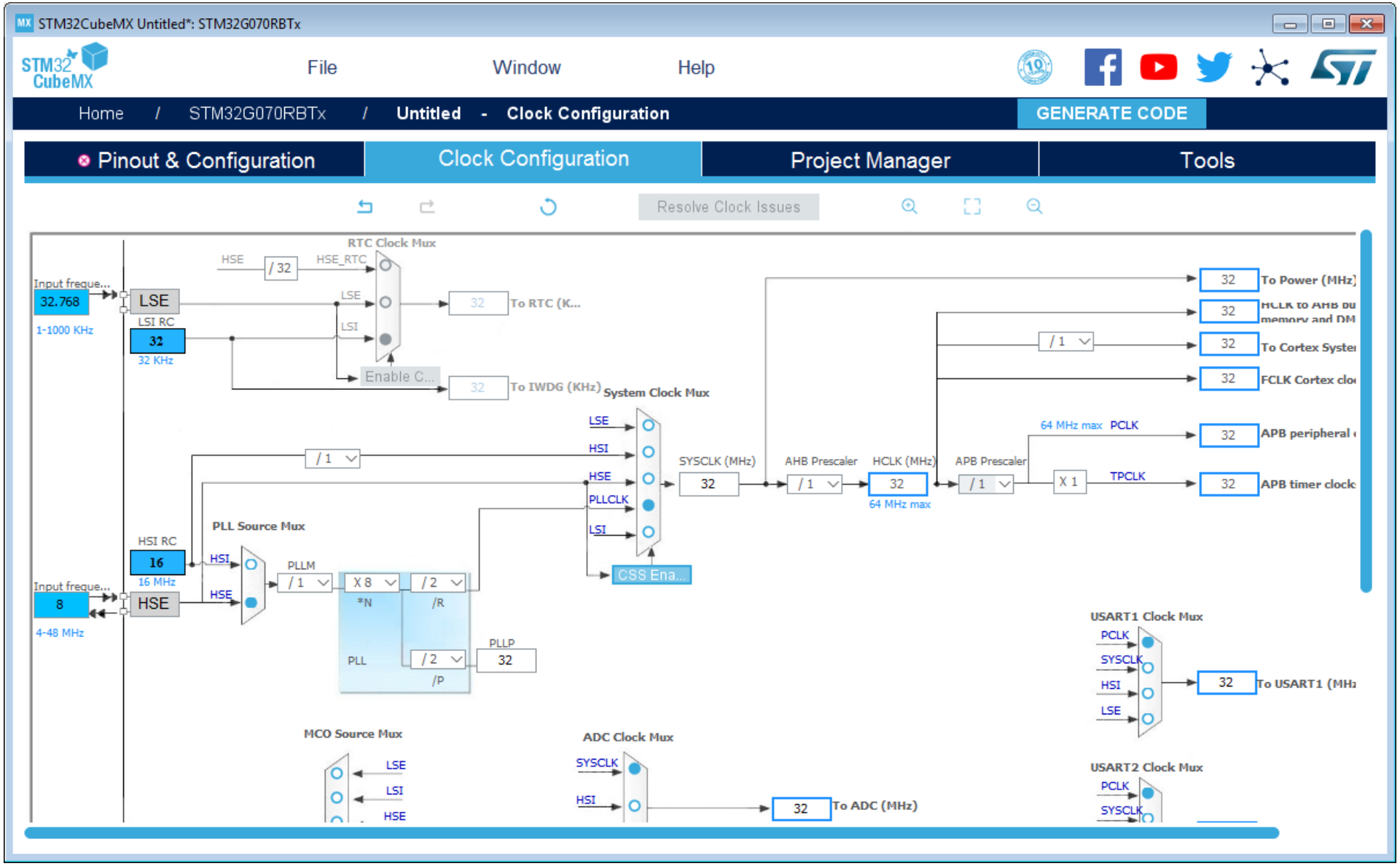
- Generates all the initialization code in C.
- Generates project file for any supported development toolchain.
- User code can be added in dedicated sections and will be kept upon regeneration.
- Option to use the latest library version or keep the same even if re-generating.



life.augmented

Clock configuration

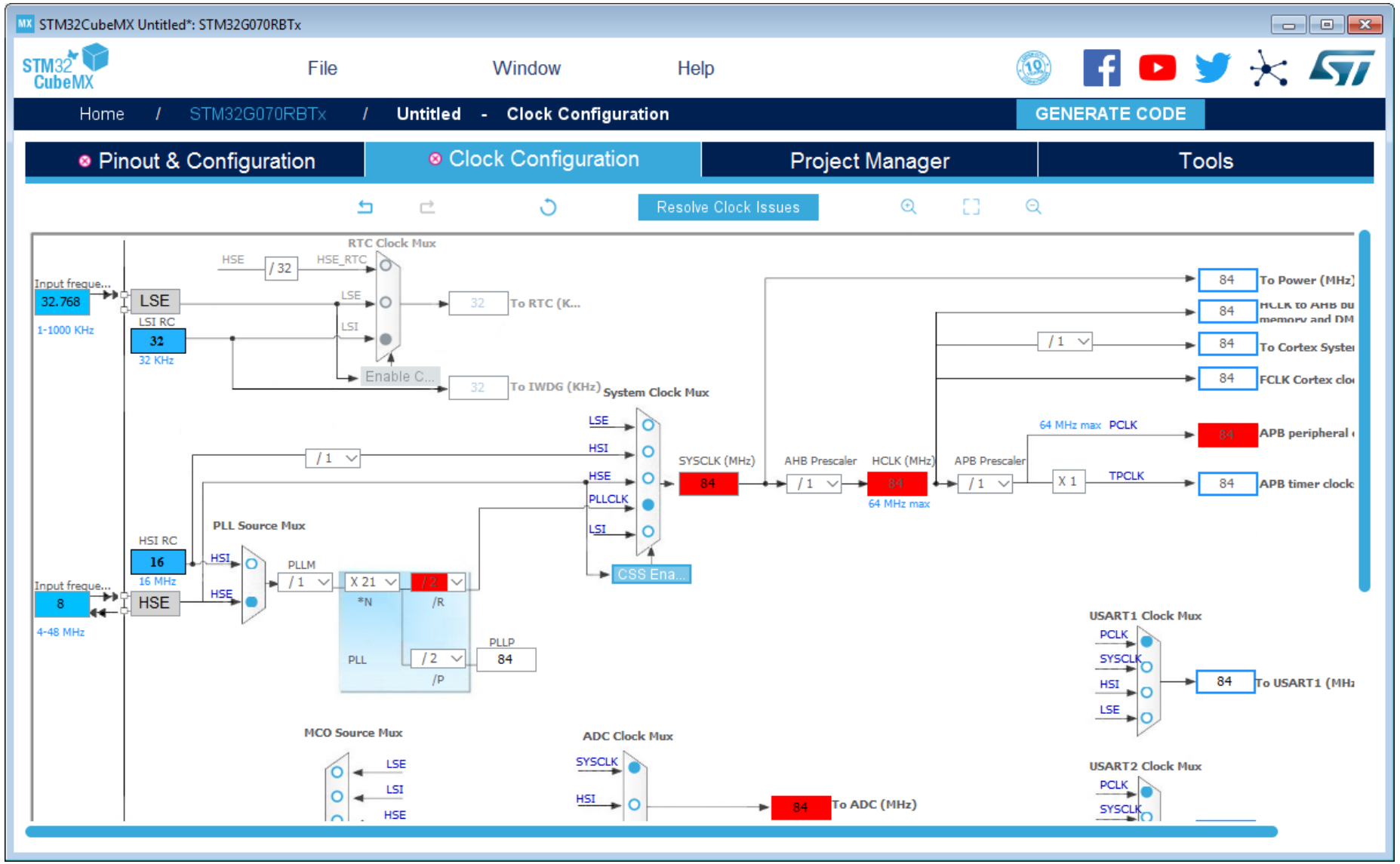
- Immediate display of all clock values.
- Active and inactive clock paths are differentiated.
- Management of clock constraints and features.



life.augmented

Clock configuration (cont.)

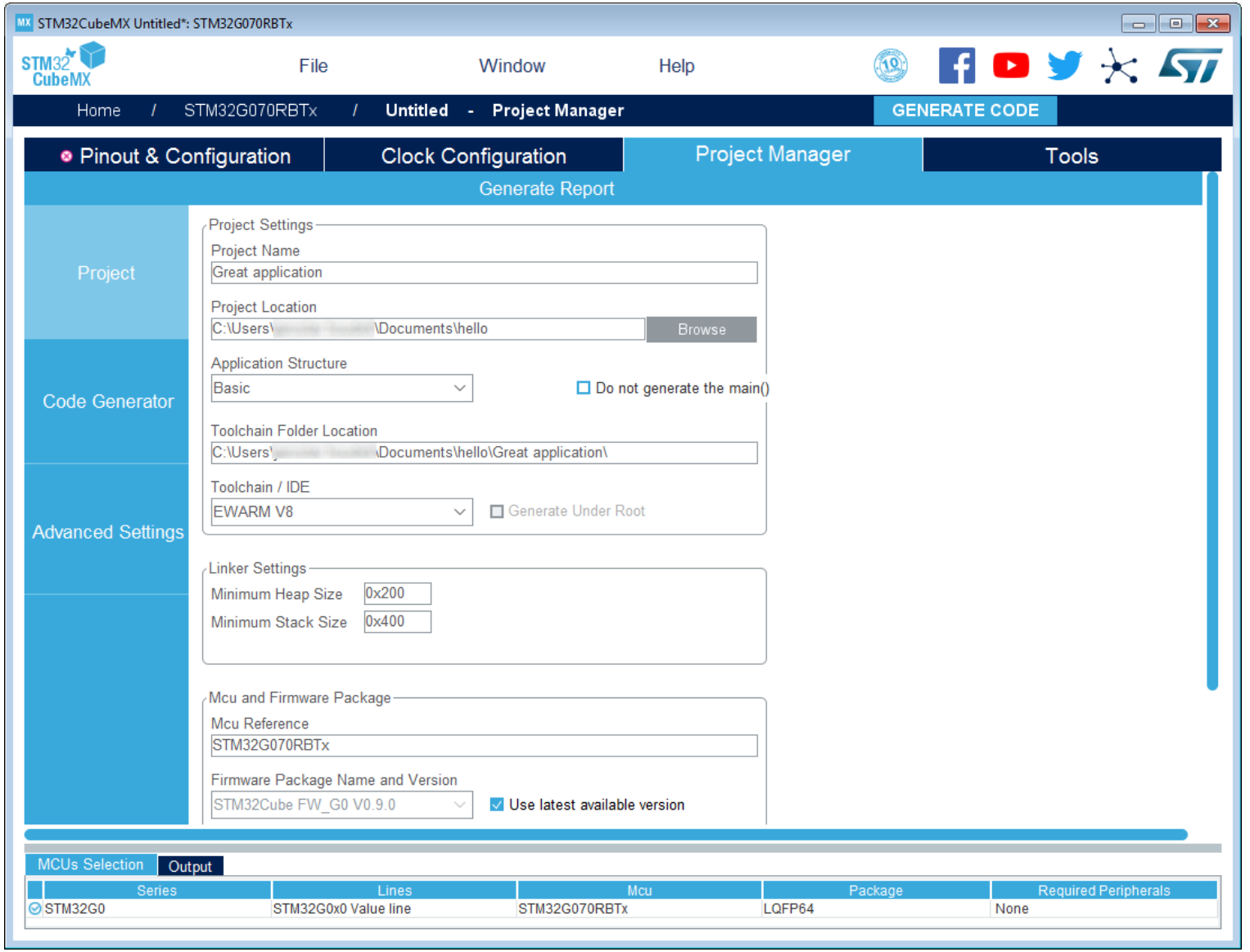
- Highlight of errors – instantly turns red.
- Enter the value in the blue frame and let the tool adjust the dividers and multipliers.
- Lock a value to prevent the tool from modifying it.



life.augmented

Code generation project settings

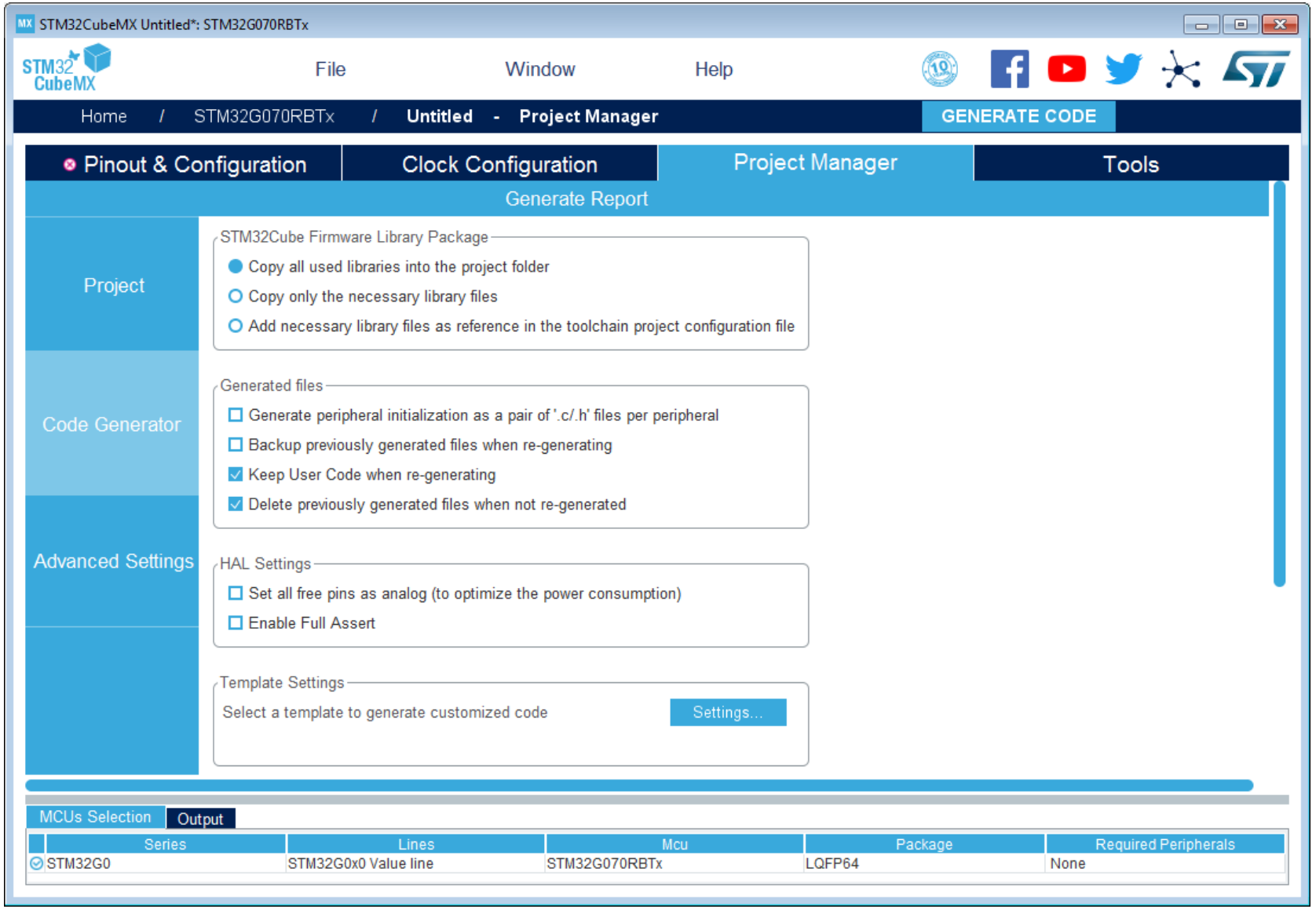
- Name your project when saving
- Browse for project location
- Pick the preferred toolchain
- Review the exact MCU type and library version



life.augmented

Code generation options

- Library package
  - Whole library or the necessary part may be copied to the generated project folder.
  - Or keep the library in original place and refer to it from all projects.
- Generated files
  - Each peripheral initialized in separate file or in common source file.
  - Options for working with old files.
  - **The option to keep user code intact is here.**
- HAL settings
  - Setting available pins to analog reduces power consumption, but be careful to **explicitly select SWD/JTAG in pinout.**
  - Full assert is useful for debugging.



life.augmented

Warning and disclaimer

- Designed for use with the entire STM32 family of microcontrollers, at times the STM32CubeMX GUI tool is unable to focus on a specific feature of a particular device.

- **The STM32CubeMX GUI tool is not a replacement for the reference manual or datasheet**
  - Always refer to written documentation for further information!
  - Important features are often available on the product or in the HAL but not in the GUI.

- The GUI helps to start a project and to initialize a working starting configuration – but the configuration can be dynamically changed at runtime (i.e. GPIO, NVIC priority or clock settings).

Hands-On #1

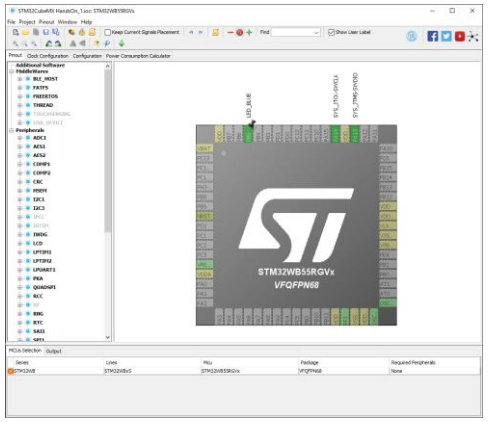
Let’s blink an LED

life.augmented

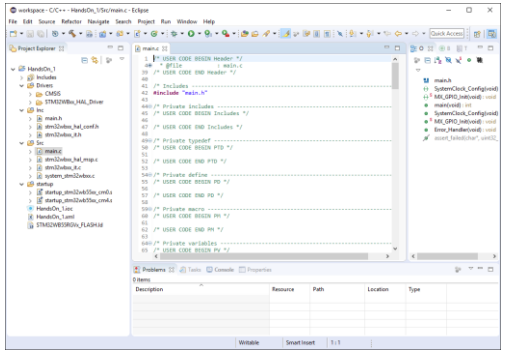
Hands-On principle

- Start to play with STM32WB and its development tools
- Experience that even though STM32WB is dual core and radio device, it is still an easy to use STM32 MCU

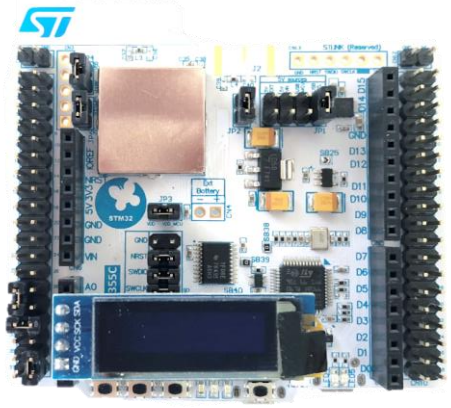
STM32CubeMX



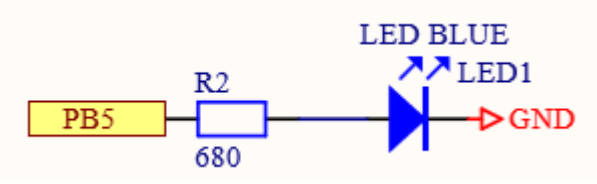
AtollicTrueSTUDIO



P-NUCLEO-WB55



Toggle @1Hz rate

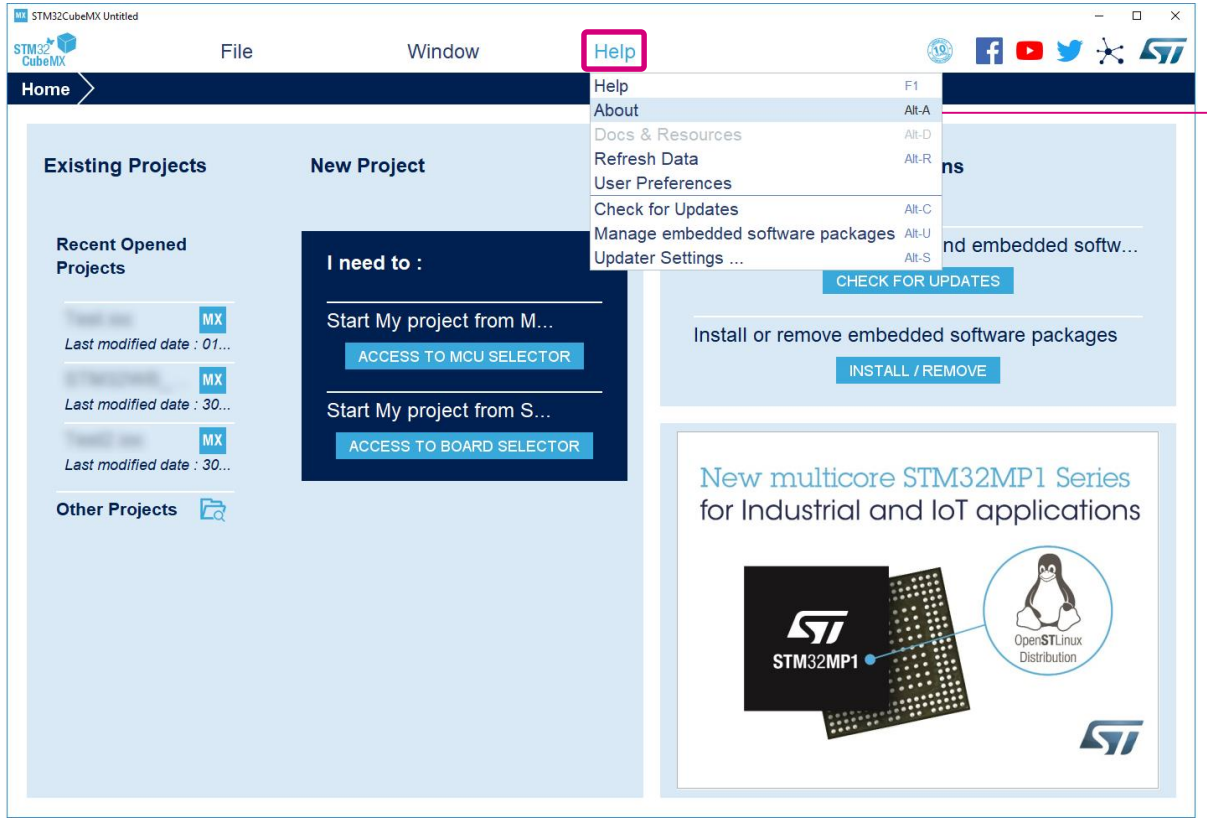


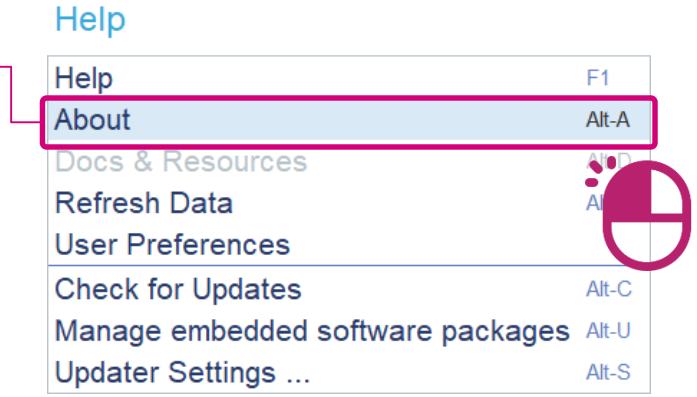
life.augmented

Open STM32CubeMX

1

MX STM32CubeMX
Desktop app

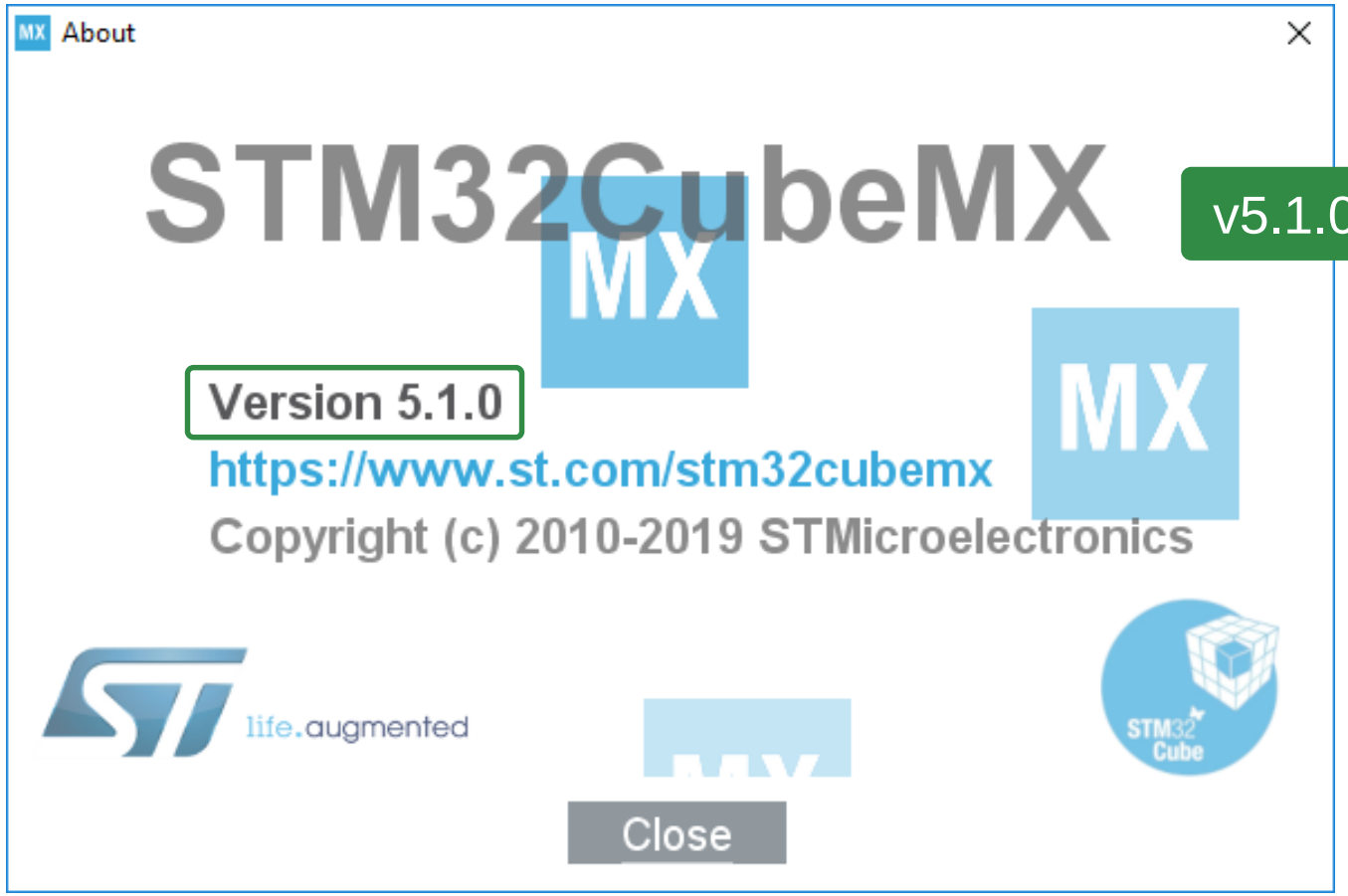




Help → About
[Alt+A]

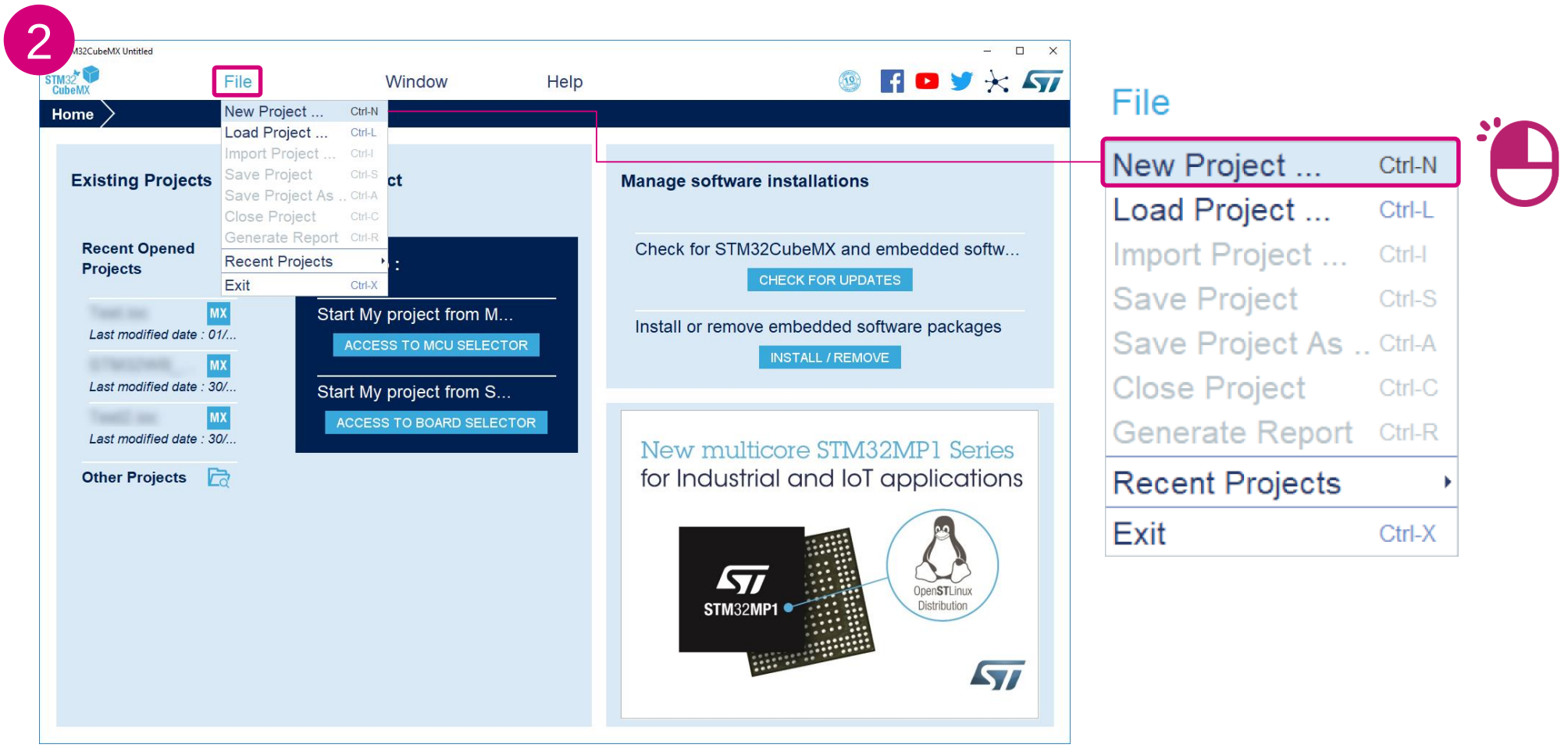
life.augmented

Check the STM32CubeMX version



v5.1.0 or newer

STM32CubeMX V5.1.0 is the first official release supporting STM32WB device



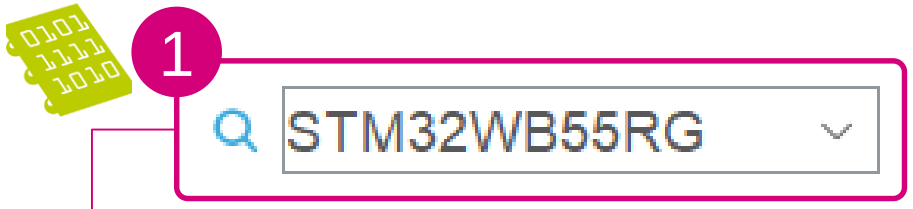
File → New Project…

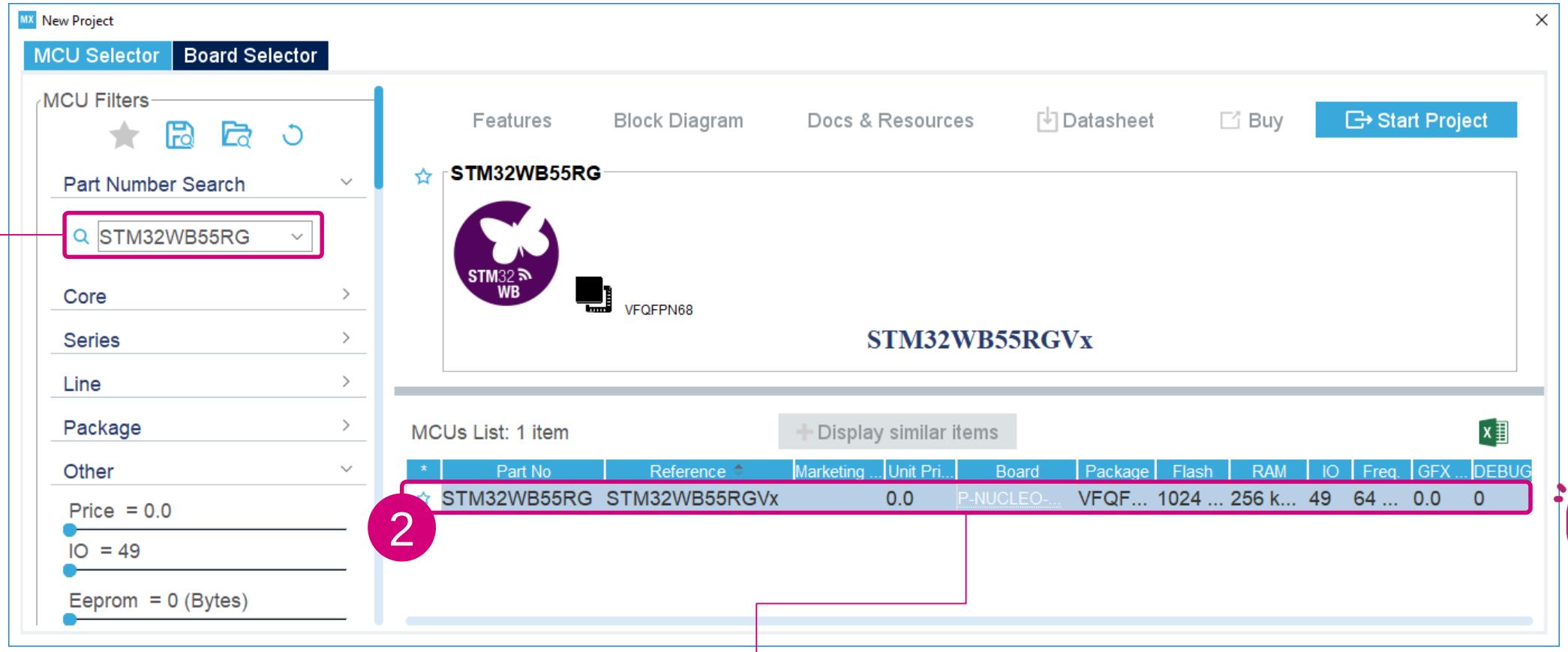
[Ctrl+N]

(or click on ACCESS TO MCU SELECTOR)

life.augmented

Select the sales type



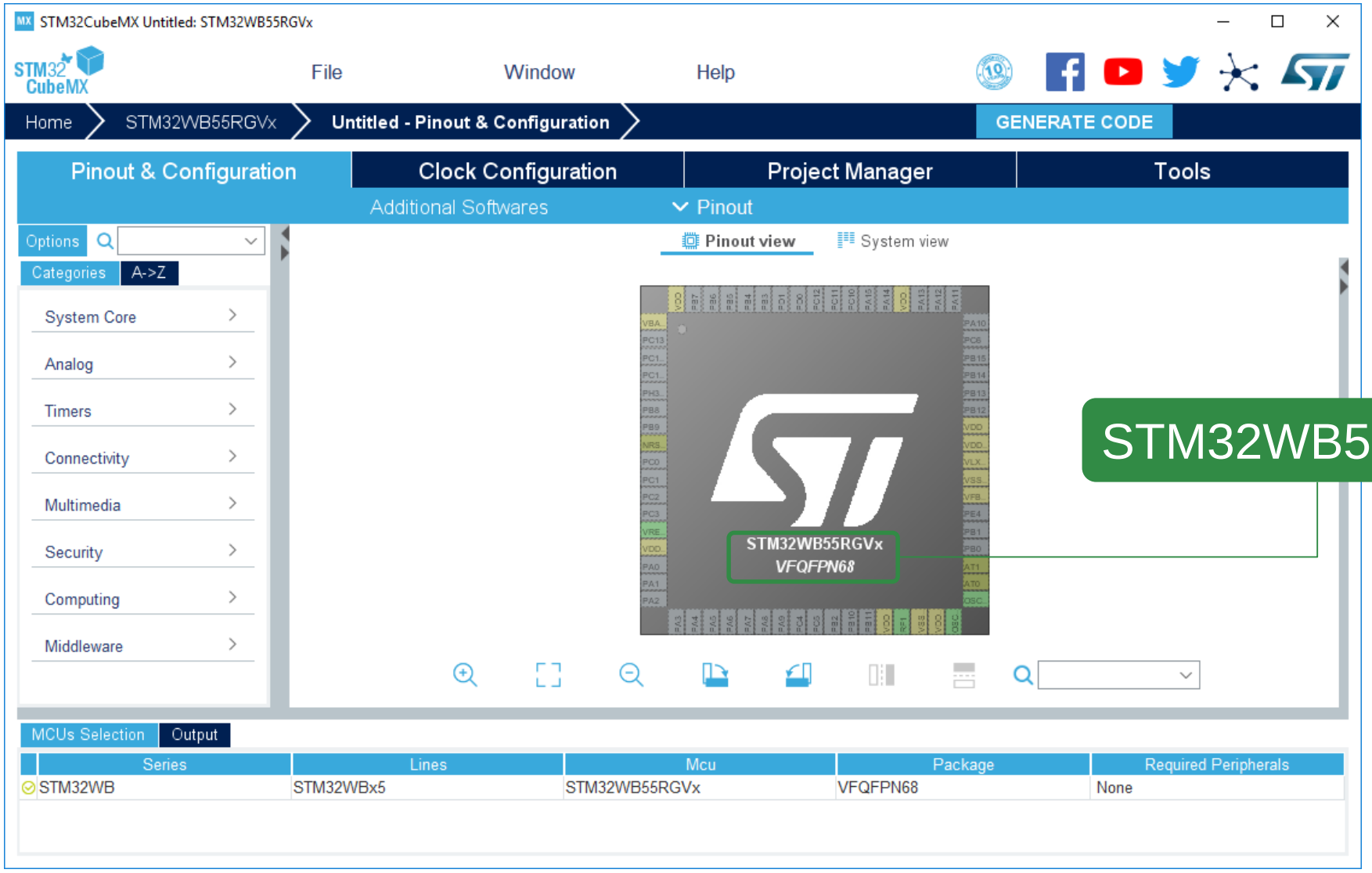


*	Part No	Reference	Marketing ...	Unit Pri...	Board	Package	Flash	RAM	IO	Freq.	GFX ...	DEBUG
☆	STM32WB55RG	STM32WB55RGVx		0.0	P-NUCLEO-...	VFQF...	1024 ...	256 k...	49	64 ...	0.0	0

*	Part No	Reference	Marketing Status	...	Board	Package	Flash	RAM	IO	Freq.	GFX Score	DEBUG
☆	STM32WB55RG	STM32WB55RGVx		...	P-NUCLEO-WB55....	VFQFPN68	1024 kBytes	256 kBytes	49	64 MHz	0.0	0

life.augmented

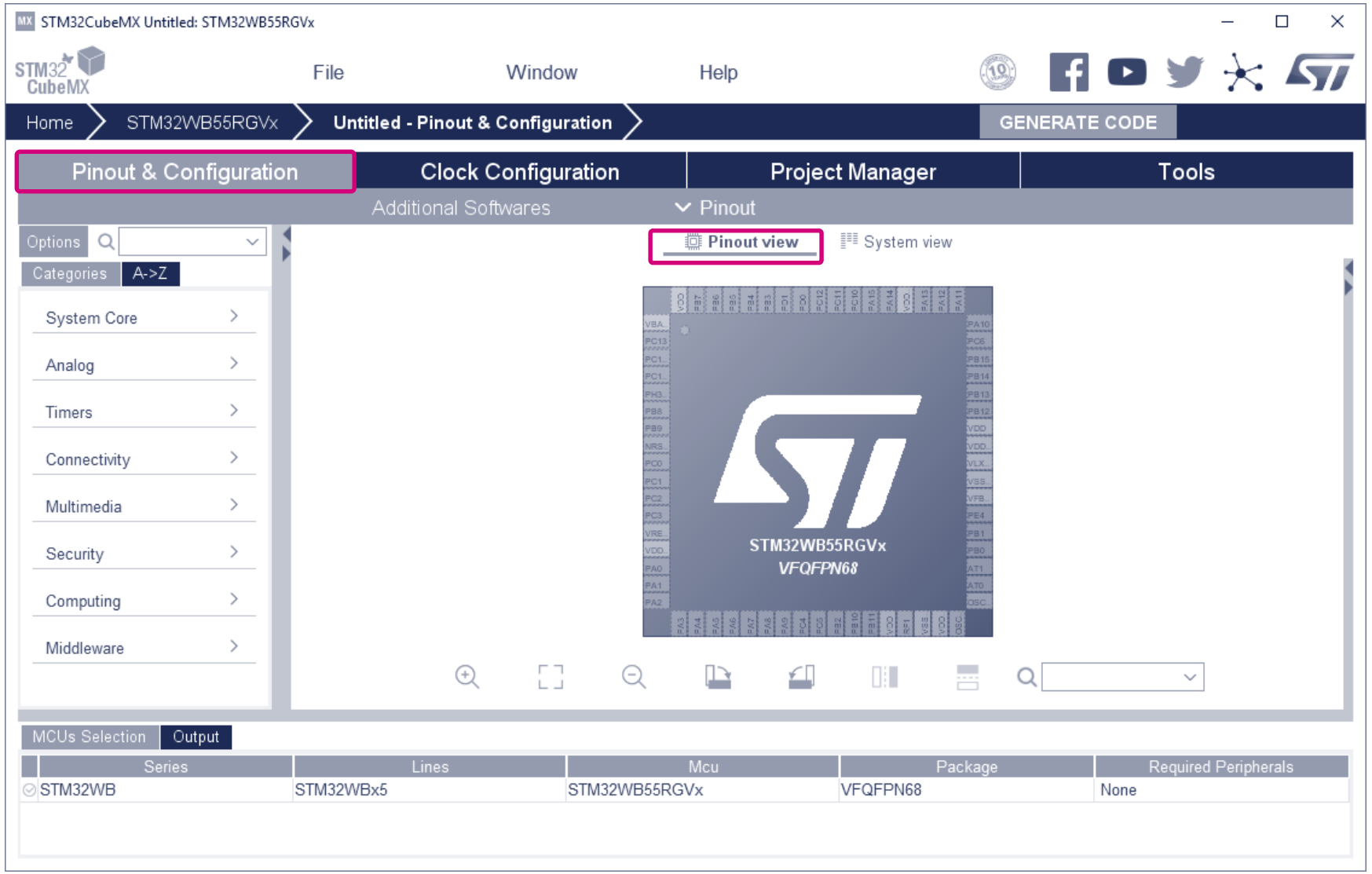
New empty project



STM32WB55RGVx

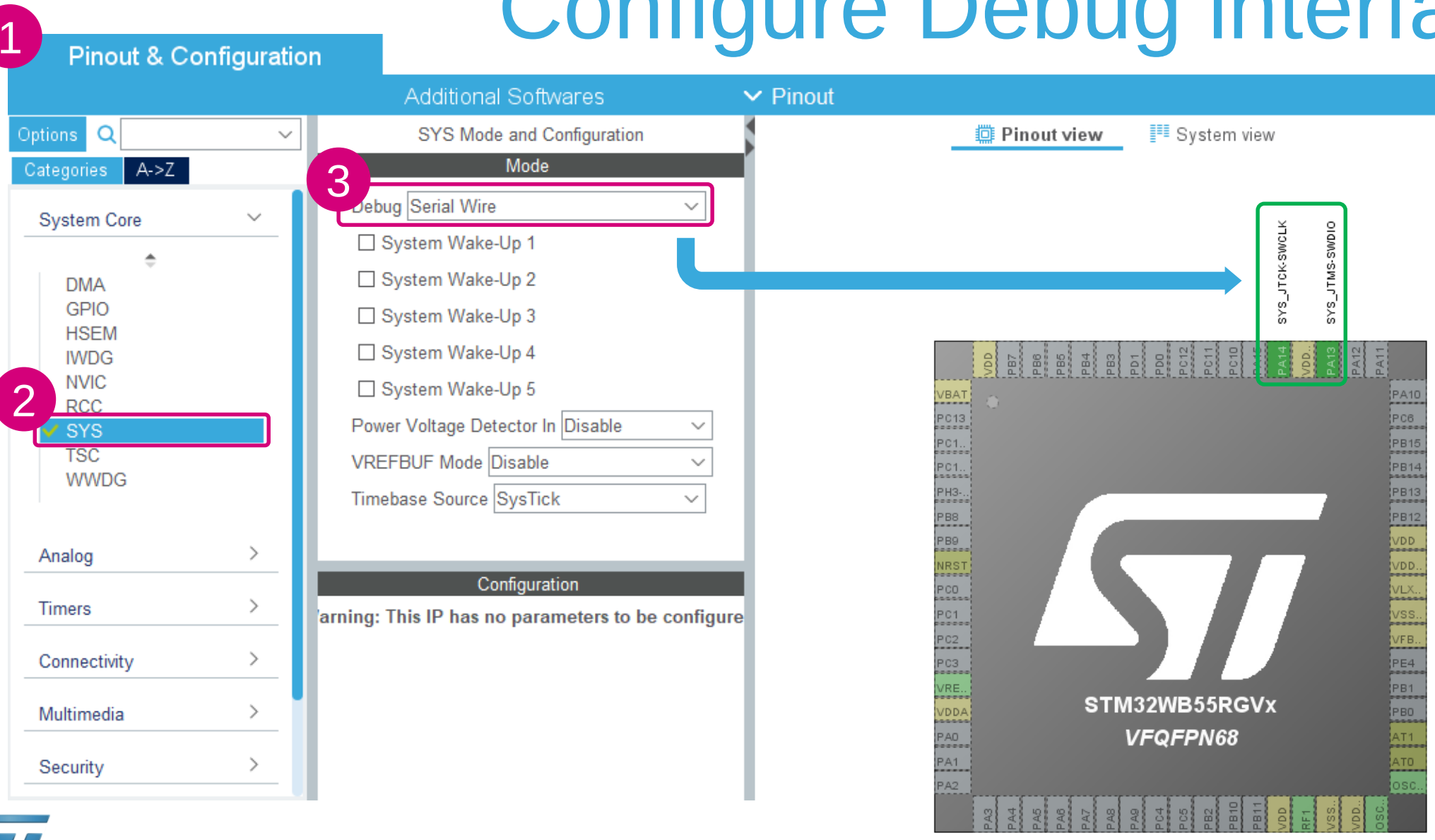
life.augmented

Let’s start with Pinout configuration

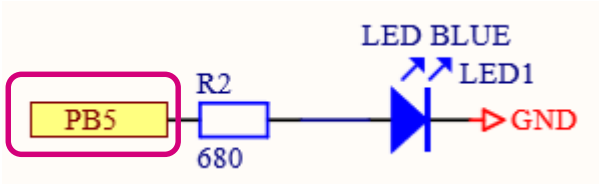


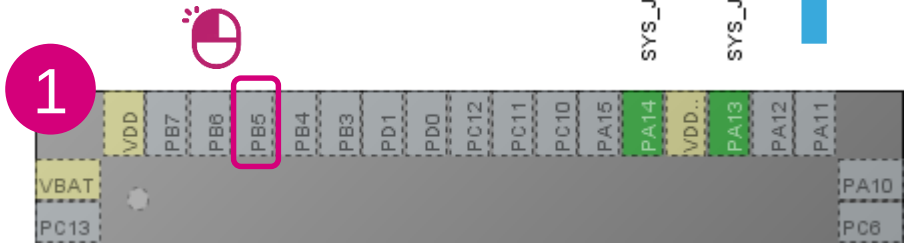
life.augmented

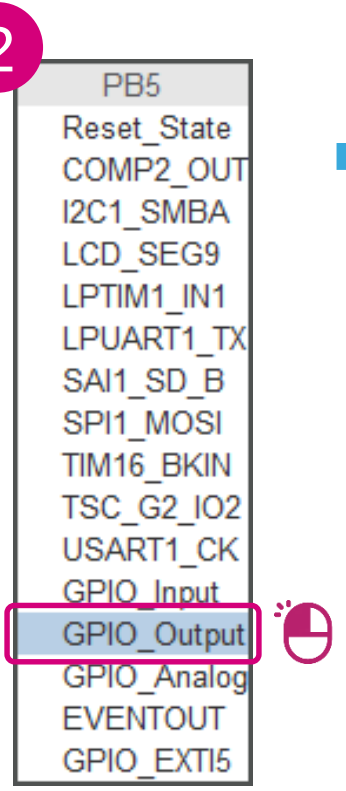
Configure Debug interface

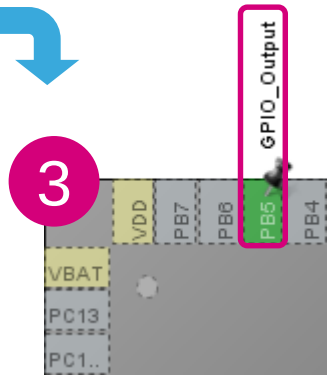


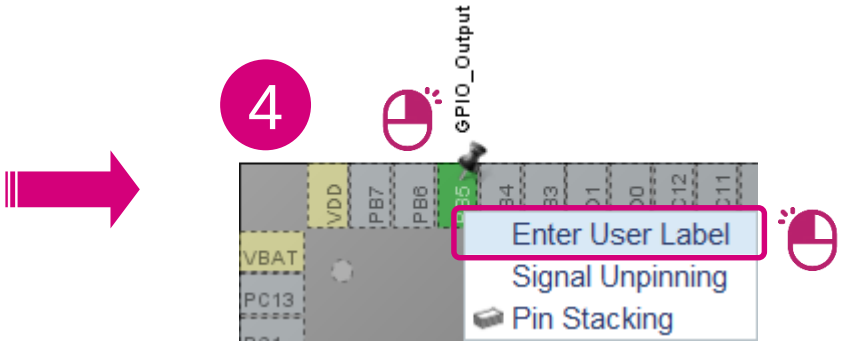
Configure the pin for blue LED control

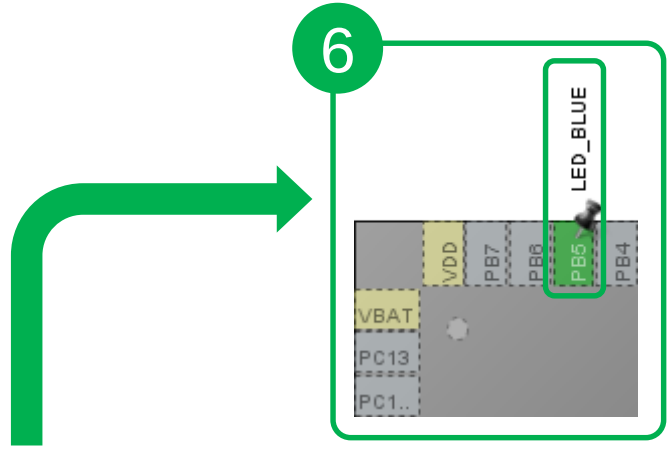








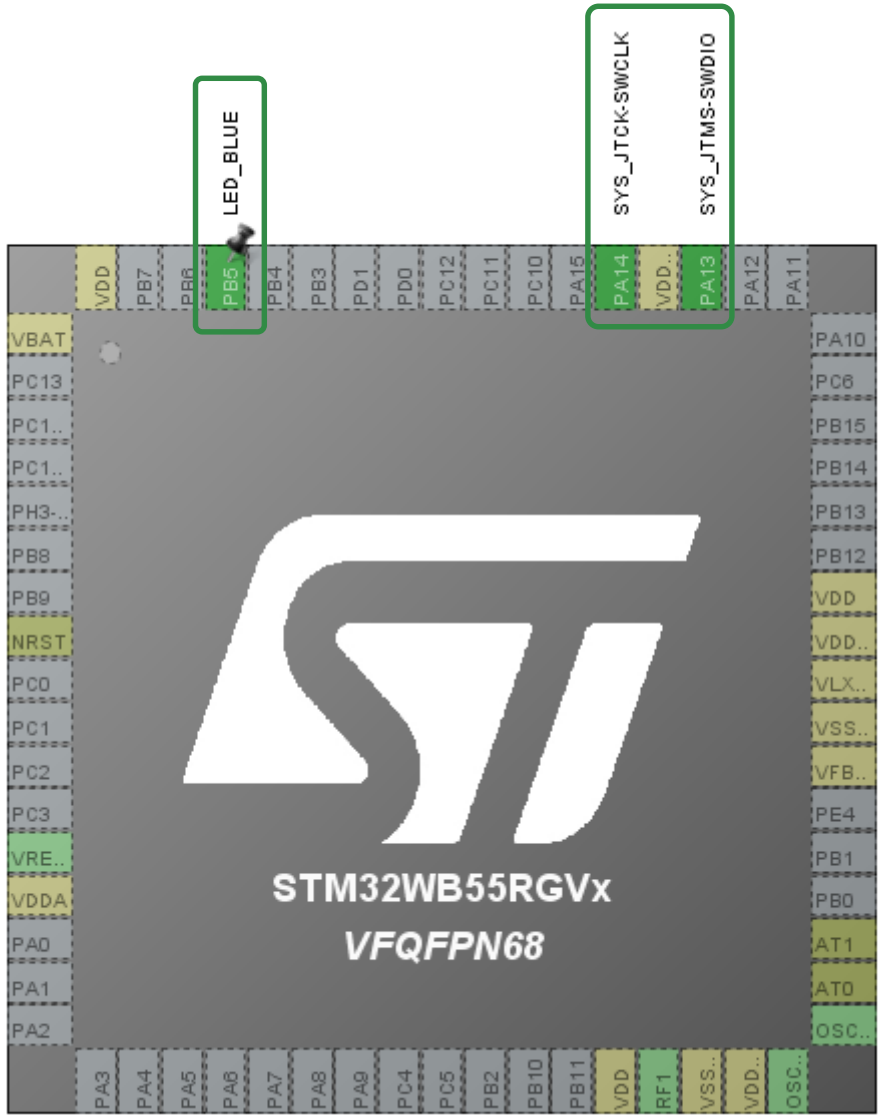




life.augmented

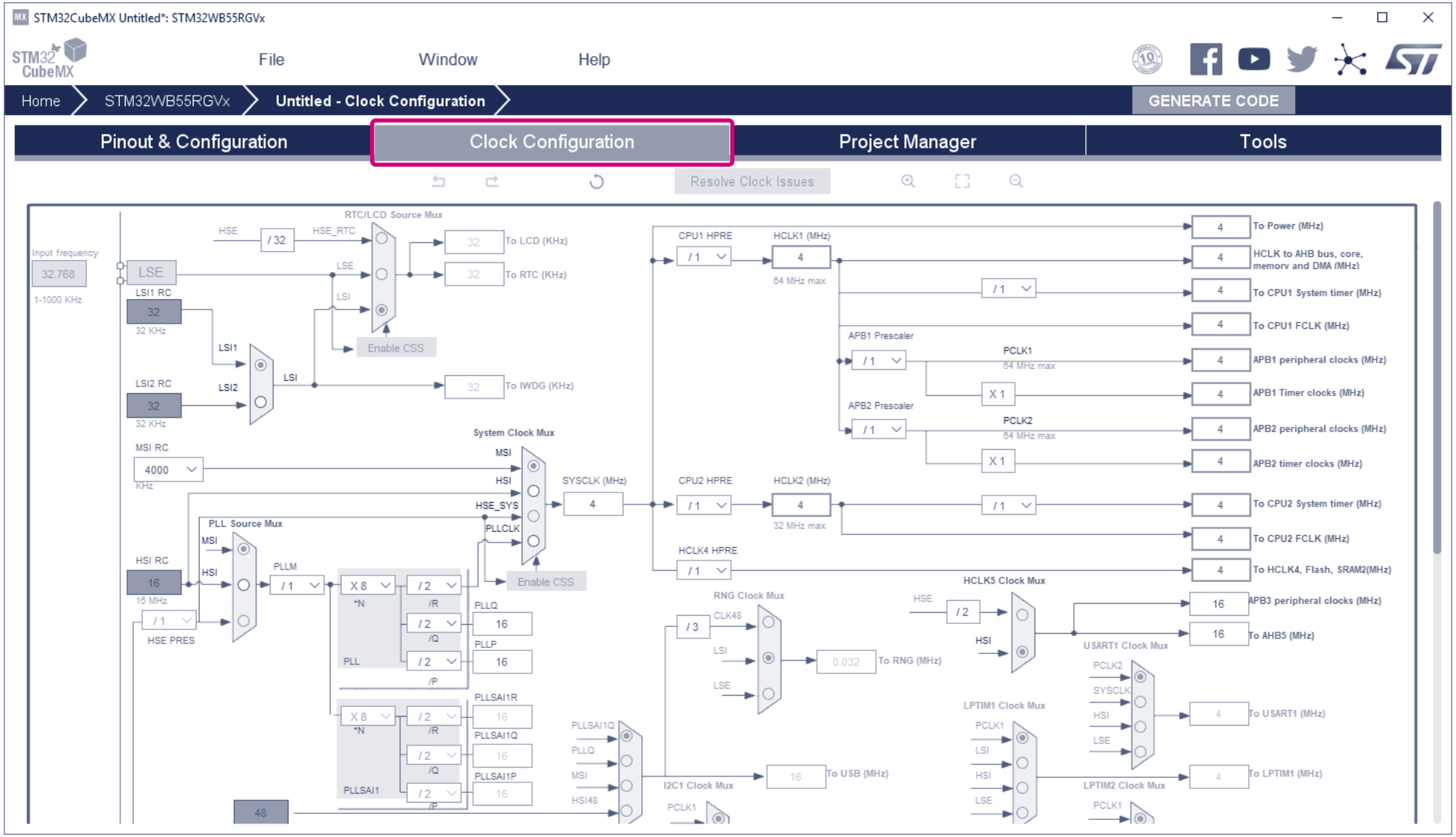
Checkpoint

SWCLK @PA14
SWDIO @PA13
LED_BLUE @PB5



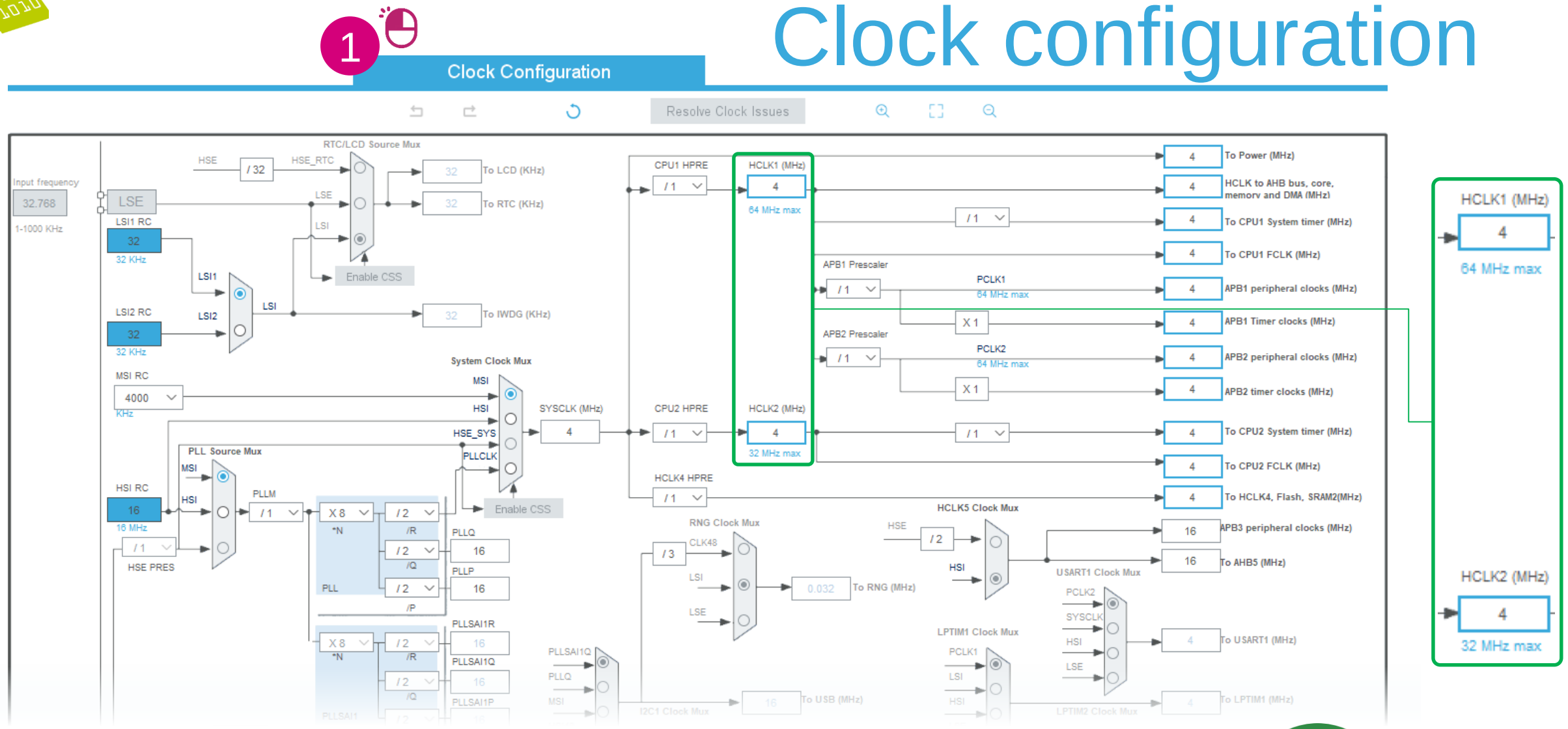
life.augmented

Now switch to clock configuration



life.augmented

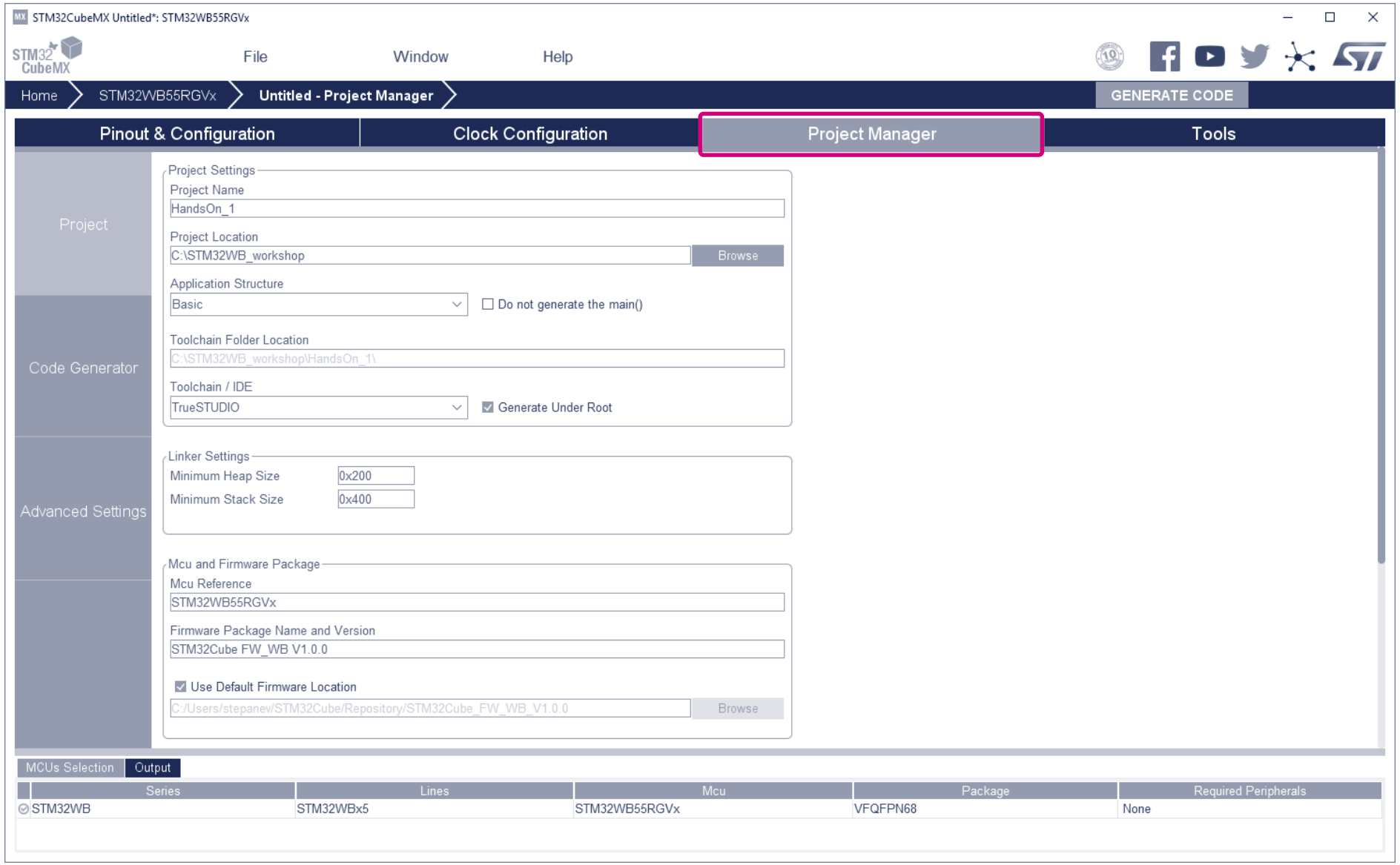
Clock configuration



Keep the default configuration – no need to change anything
(complete system @4MHz from MSI)

life.augmented

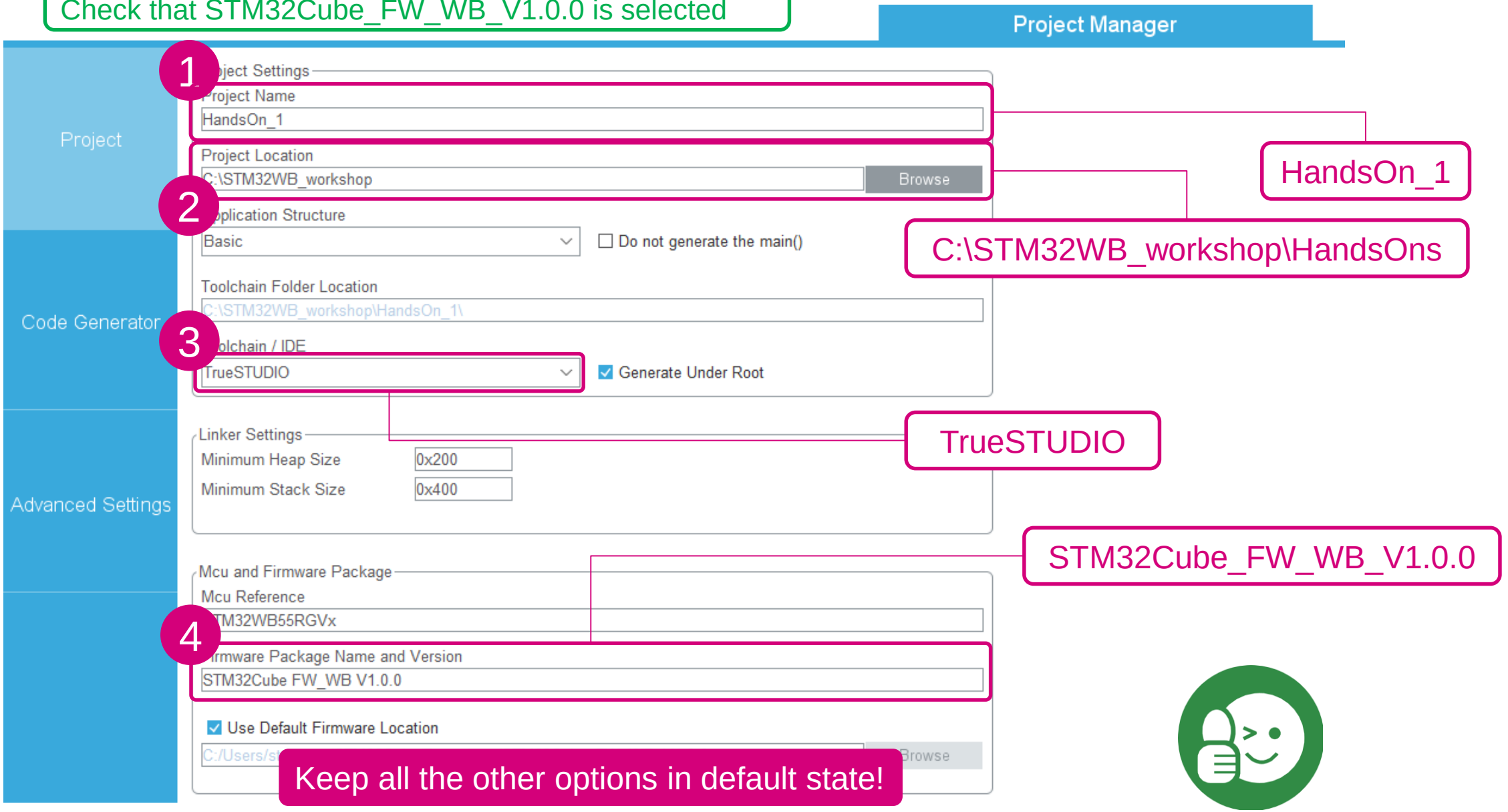
Finalize the project settings



life.augmented

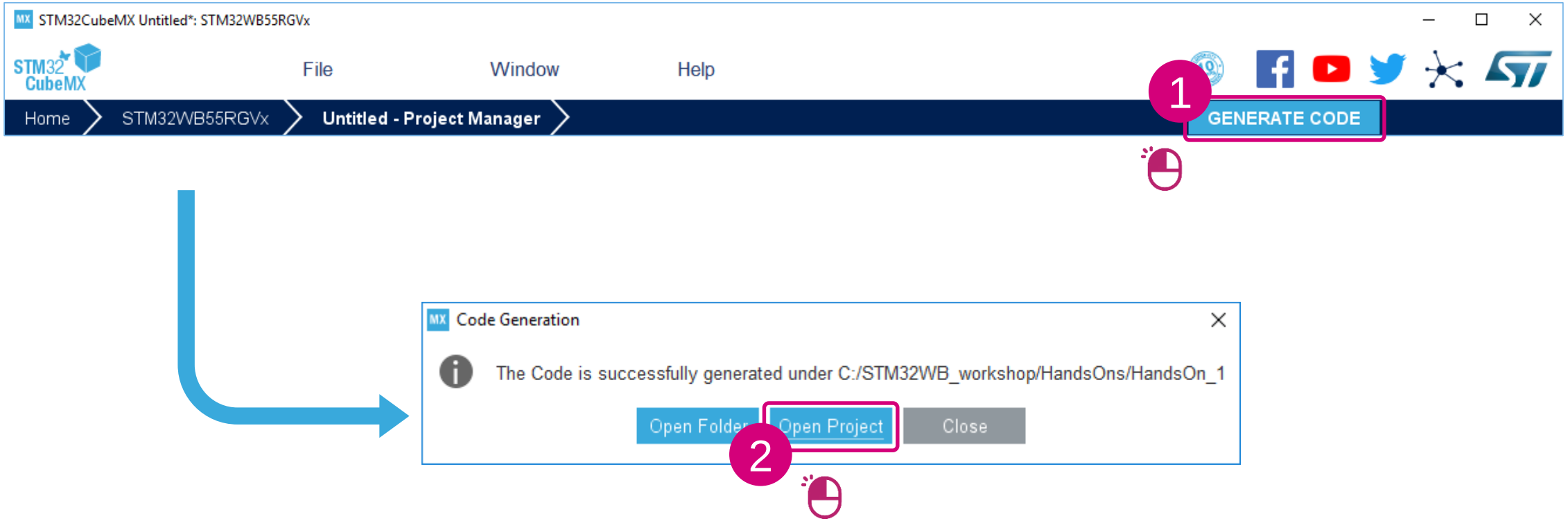
Project Name → HandsOn_1
Project Location → C:\STM32WB_workshop\HandsOns\
IDE → STM32CubeIDE or TrueSTUDIO
Check that STM32Cube_FW_WB_V1.0.0 is selected

Project settings



life.augmented

Generate the code

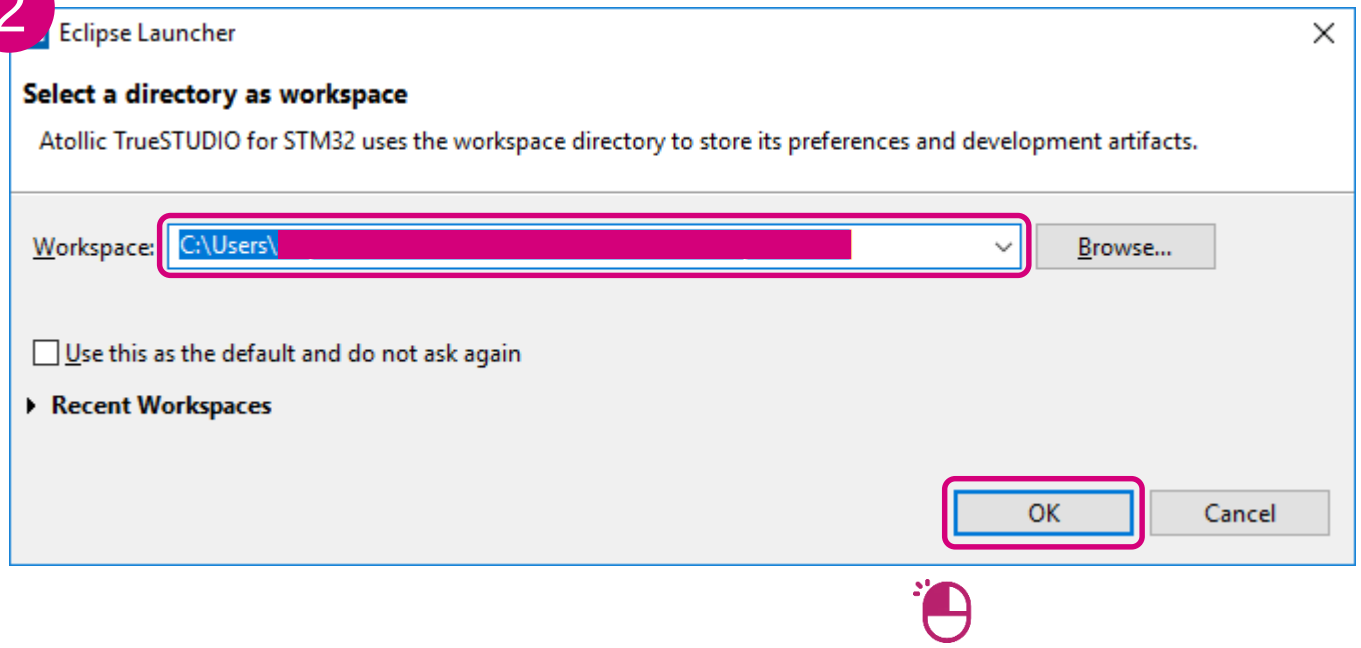


life.augmented

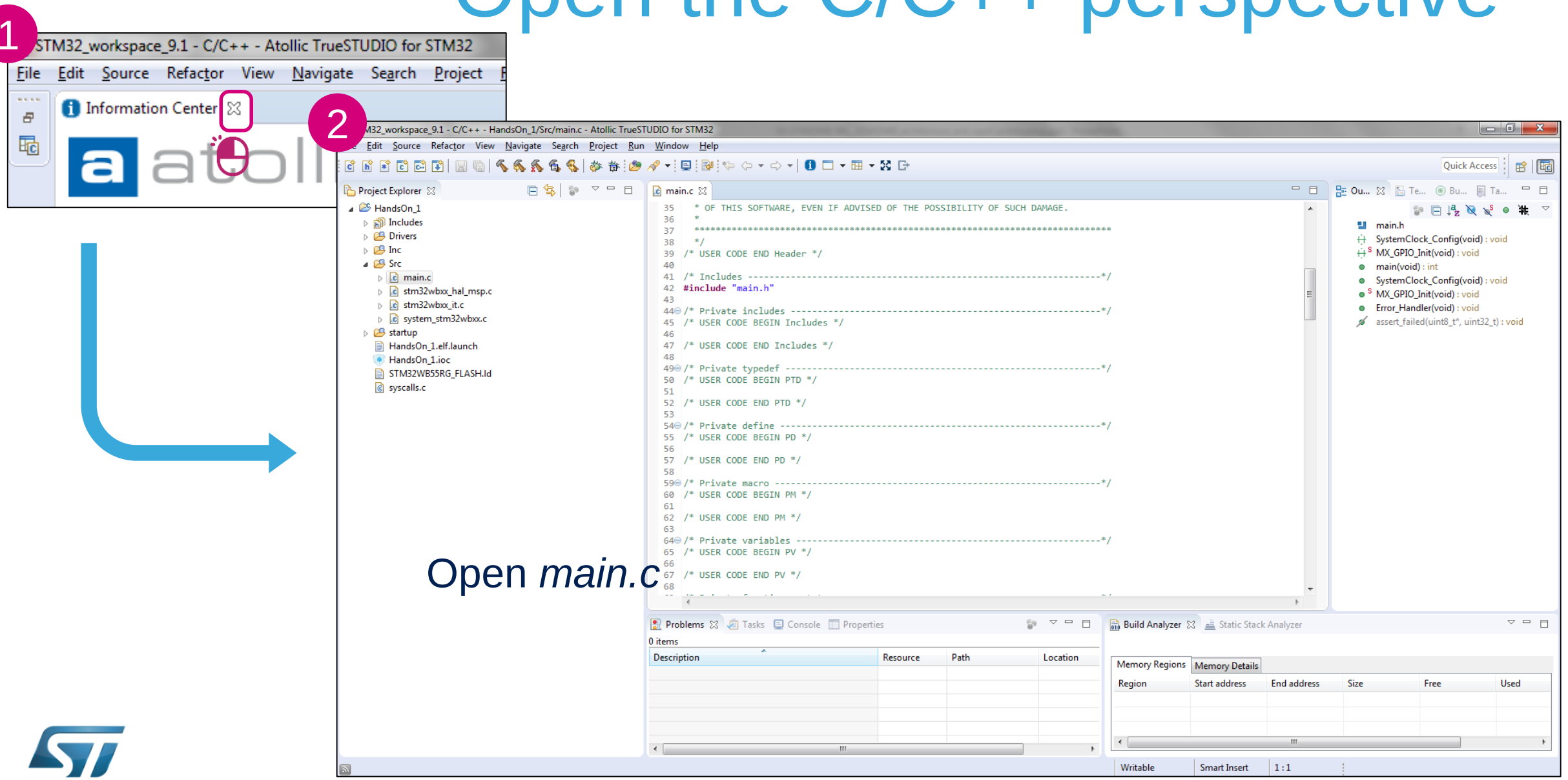
Atollic TrueStudio project opening

1

2

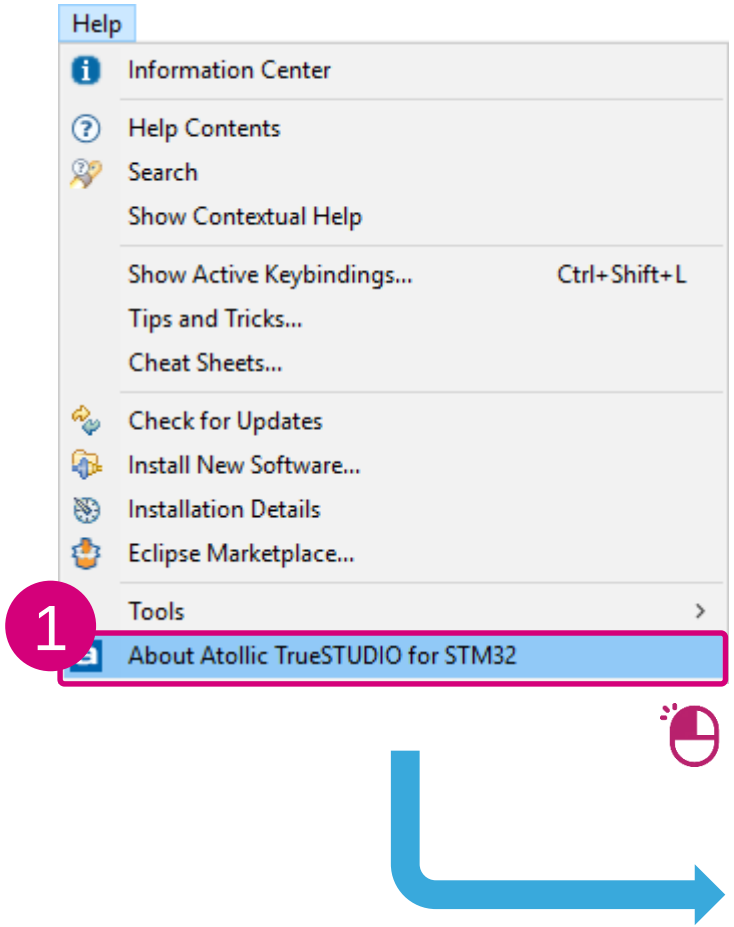


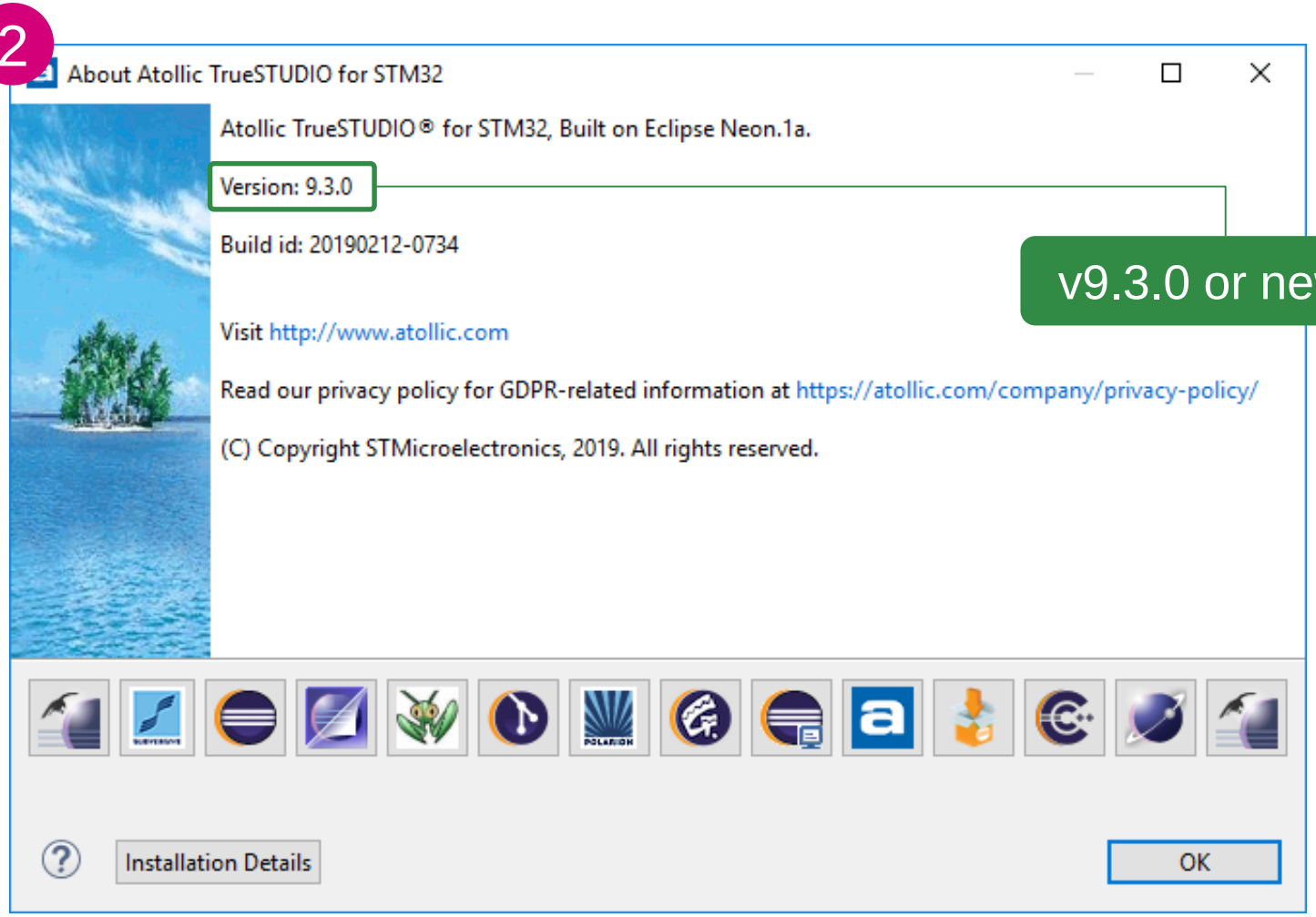
Open the C/C++ perspective



life.augmented

Check the Atollic TrueSTUDIO version





v9.3.0 or newer

Help → About Atollic TrueSTUDIO for STM32

life.augmented

Check the Atollic TrueSTUDIO version

TrueSTUDIO NRND

Print Save to MyST Share via Email

A powerful eclipse based C/C++ integrated development tool for your STM32 projects

Get Software

Overview Tools & Software

Atollic® TrueSTUDIO® for STM32 is a flexible and extensible development and debugging IDE for STM32 MCU developers who want extremely powerful tools to aid in development of high-quality embedded software. TrueSTUDIO® is based on open standards (ECLIPSE and GNU) and extended with professional features for code management and advanced system analysis. This gives a unique insight into the structure and the dynamic behavior of the system.

With TrueSTUDIO®, STM32 developers get a professional development tool, built on open standards, while at the same time including advanced debug and software engineering features that help developers increase their efficiency.
In particular, TrueSTUDIO® provides STM32 developers with a wide selection of system analysis functions, helping them to assess the design soundness from a variety of angles (including static analysis on memory and stack usage, as well as visualizing the dynamic

Support & Feedback

life.augmented

Check the STM32CubeIDE version

https://www.st.com/STM32CubeIDE

STM32CubeIDE ACTIVE

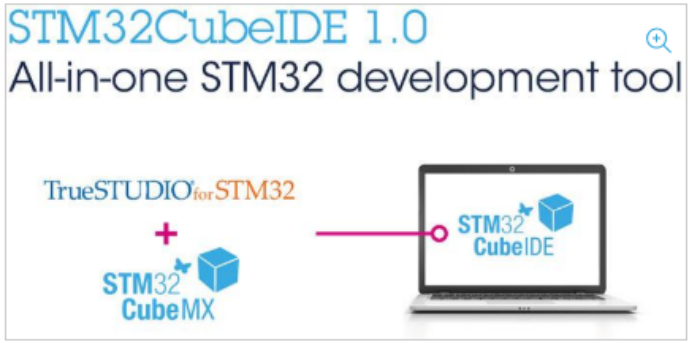
Print Save to MyST Share via Email

Integrated Development Environment for STM32

Get Software | Download databrief

Overview | Tools & Software | Resources

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem.



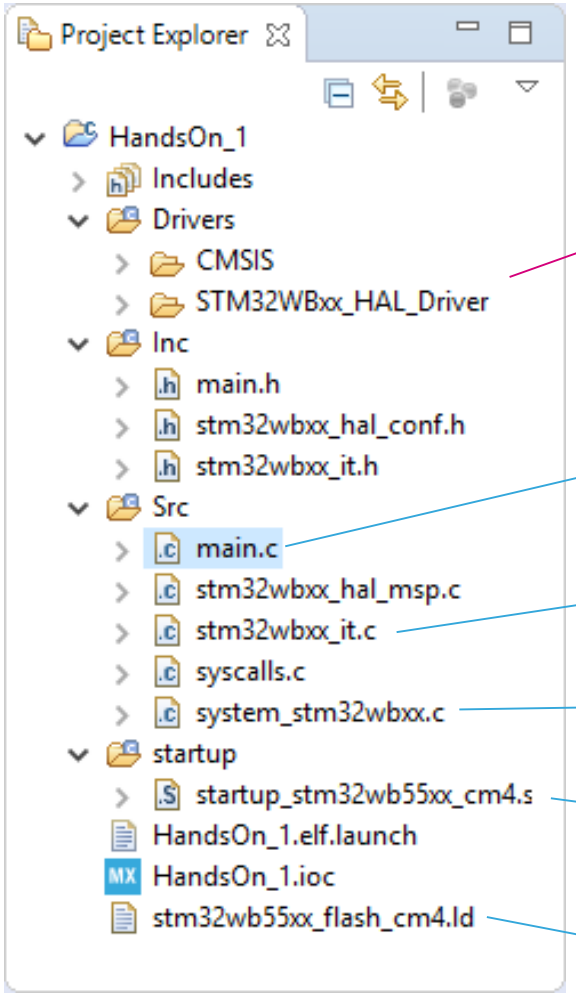
STM32CubeIDE is an advanced C/C++ development platform with IP configuration, code generation, code compilation, and debug features for STM32 microcontrollers. It is based on the ECLIPSE™/CDT framework and GCC toolchain for the development, and GDB for the debugging. It allows the integration of the hundreds of existing plugins that complete the features of the ECLIPSE™ IDE.

STM32CubeIDE integrates all STM32CubeMX functionalities to offer all-in-one tool experience and save installation and development time. After the selection of an empty STM32 MCU or preconfigured microcontroller from the selection of a board, the project is created and initialization code generated. At any time during the development, the user can return to the initialization and configuration of the IPs or middleware and regenerate the initialization code

Support & Feedback

life.augmented

Check out the project tree



ARM CMSIS library
STM32WBxx HAL drivers

Main

Interrupt handlers

Pre-main system configuration

Startup file

Linker file

life.augmented

Add the code to toggle the LED_BLUE pin

main.c main() while(1) {} loop - user code section { **3** }

```
/* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_GPIO_TogglePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin);
    HAL_Delay(1000);
  }
  /* USER CODE END 3 */
```

toggle_LED.txt

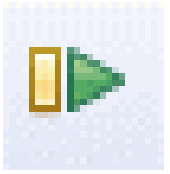
life.augmented

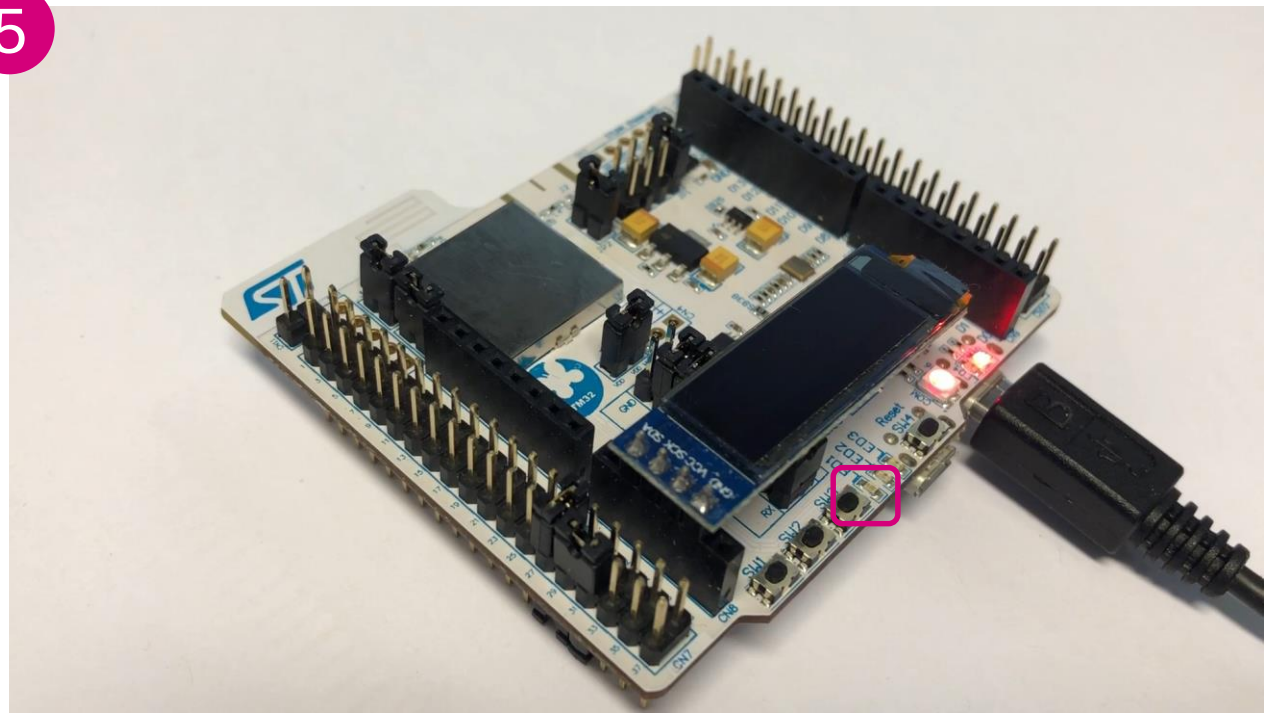
Download the app to the device

1. Connect nucleo board to PC

2. 
Build (CTRL+B)

3. 
Debug (F11)

4. 
Resume (F8)

5. 

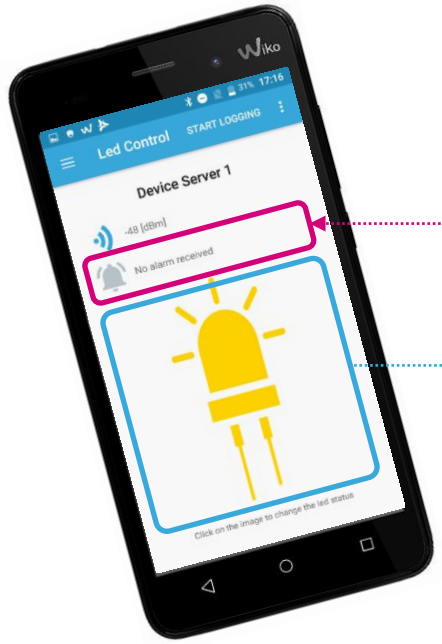
life.augmented

Hands-On #2

life.augmented

Target

Central **GATT Client**

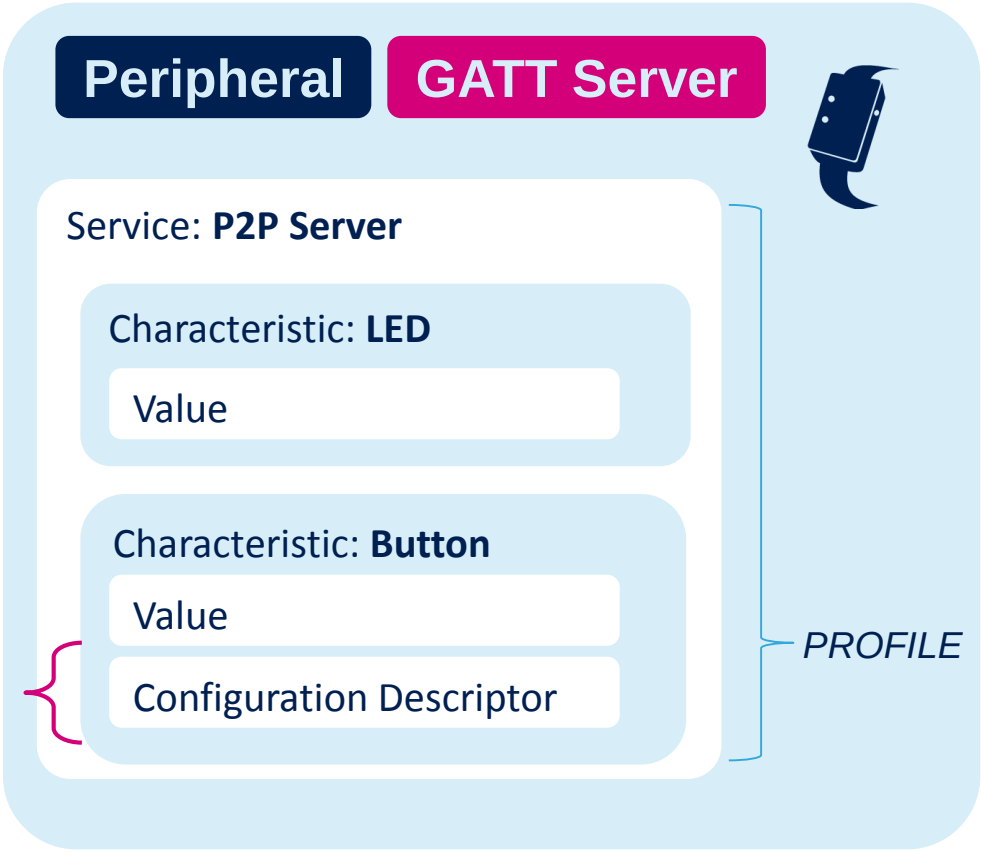


Peripheral **GATT Server**

life.augmented

Bluetooth LE basic principles

GATT – Generic Attribute Profile



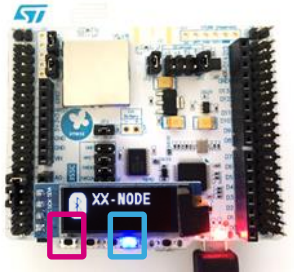
Used in this case by client to enable notification on Button characteristic value change

How should it work?

Central **GATT Client**

P2P Client **P2P Server**

Peripheral **GATT Server**



Scanning Mode

Advertising Mode

Advertising data

Advertising data

Connect Request

Link (connection) established

ATT Handle Discovery

GATT Procedure establishment

ATT Services & Characteristics

Enable Notification

Central-to-Peripheral communication **(Control SLAVE)**

Write (0x01) – Led ON

Write (0x00) – Led OFF

Tap Button in the phone app to write new characteristic #2 value

Push SW1 to update characteristic #1 value (notify value change)

Notification (0x00) – 1st Button press

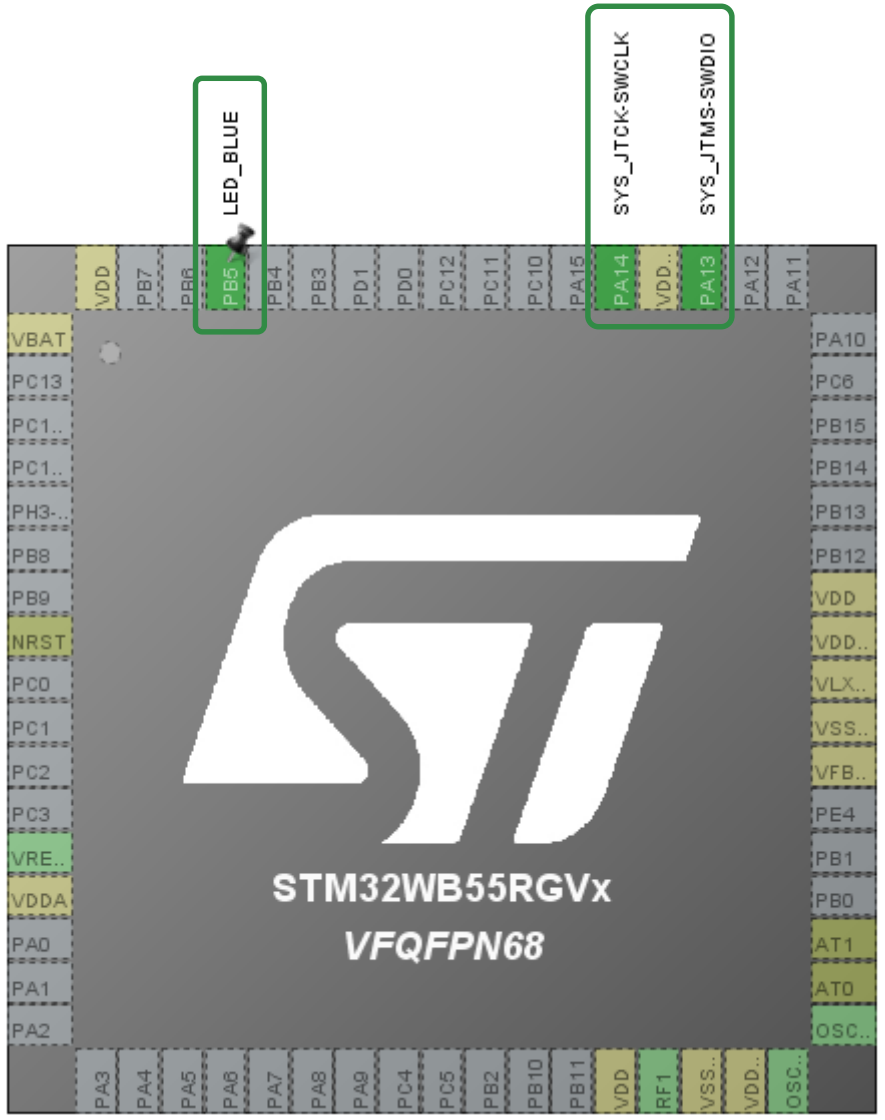
Notification (0x00) – 2nd Button press

Peripheral-to-Central communication **(Notify MASTER)**

Let's start with STM32CubeMX

life.augmented

Starting point



SWCLK @PA14
SWDIO @PA13
LED_BLUE @PB5

HandsOn_1

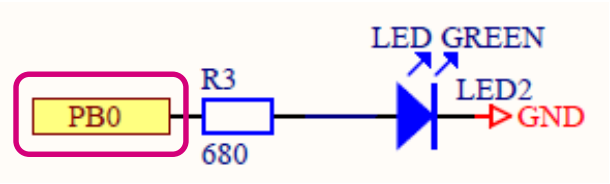
Let’s do a small time shift!

Move to slide **10**!!!

life.augmented

SKIPPED SLIDE

Add pin for green LED control

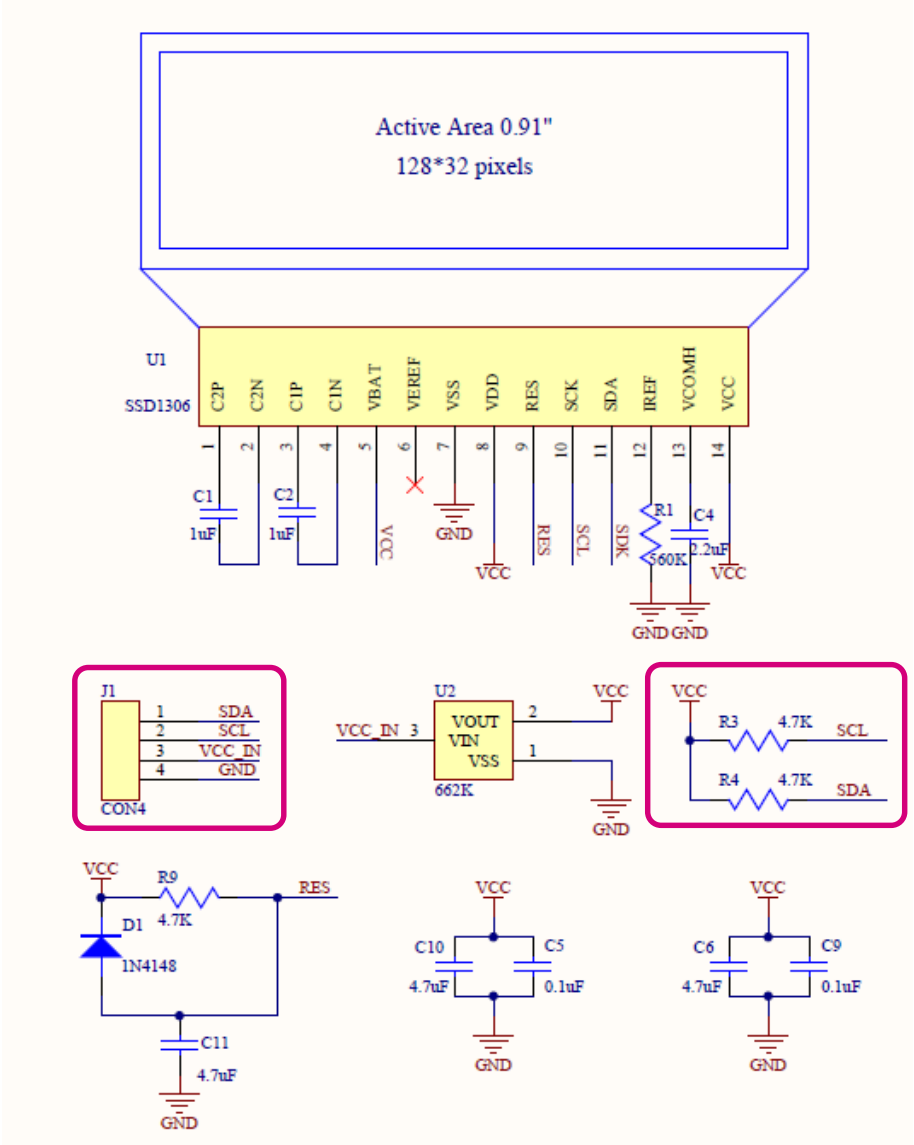


Just for development and debugging purposes.

life.augmented

SKIPPED SLIDE

OLED display connection

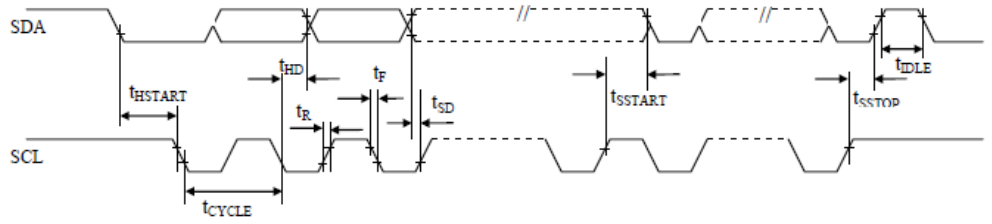


SSD1306 datasheet

Table 13-6 :I²C Interface Timing Characteristics

Symbol	Parameter	Min	Typ	Max	Unit
t_{cycle}	Clock Cycle Time	2.5	-	-	us
t_{HSTART}	Start condition Hold Time	0.6	-	-	us
t_{HD}	Data Hold Time (for "SDA_{OUT}" pin)	0	-	-	ns
	Data Hold Time (for "SDA_{IN}" pin)	300	-	-	ns
t_{SD}	Data Setup Time	100	-	-	ns
t_{SSTART}	Start condition Setup Time (Only relevant for a repeated Start condition)	0.6	-	-	us
t_{SSTOP}	Stop condition Setup Time	0.6	-	-	us
t_R	Rise Time for data and clock pin	-	-	300	ns
t_F	Fall Time for data and clock pin	-	-	300	ns
t_{IDLE}	Idle Time before a new transmission can start	1.3	-	-	us

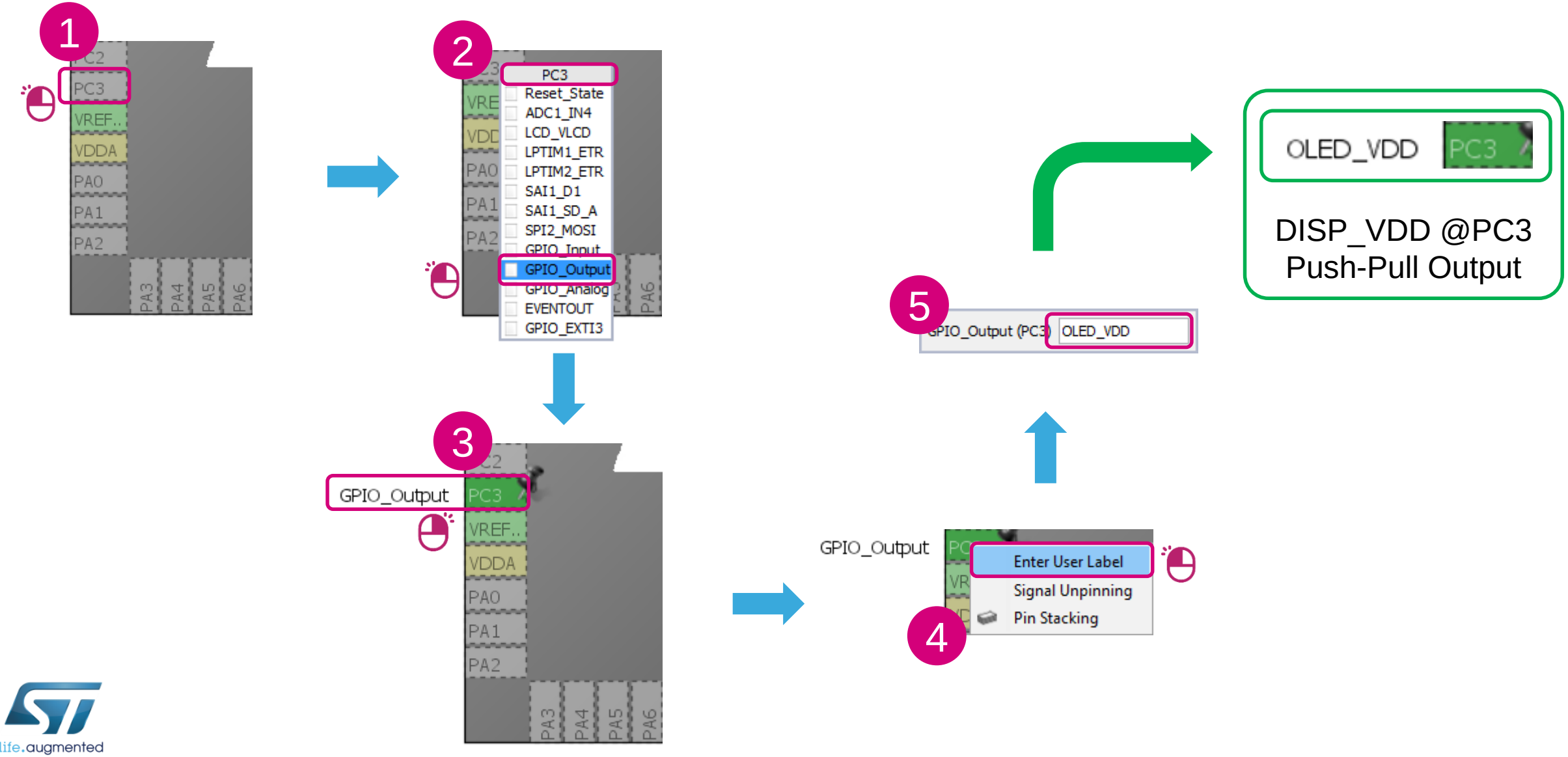
Figure 13-5 : I²C interface Timing characteristics



life.augmented

SKIPPED SLIDE

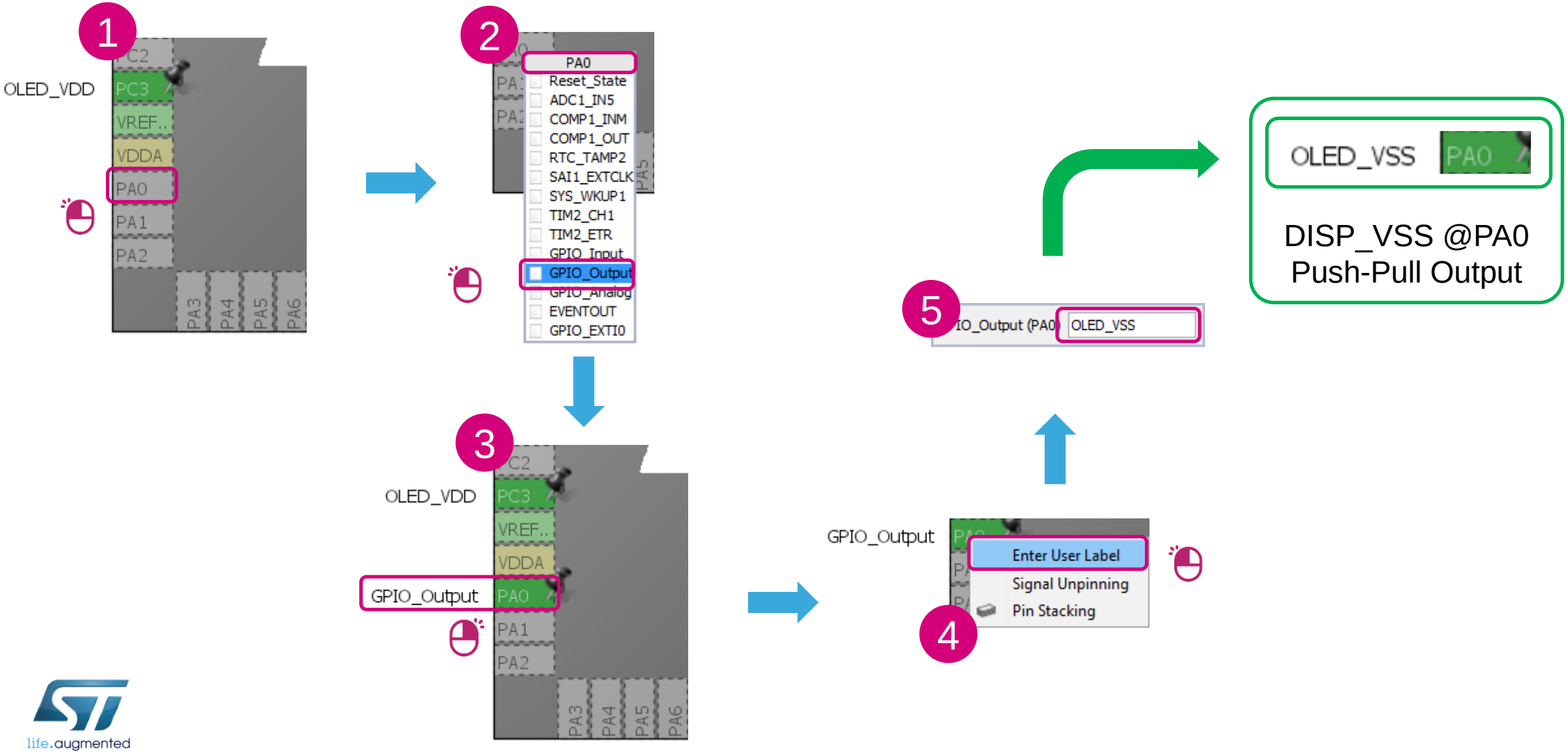
Add pin for OLED display VDD



life.augmented

SKIPPED SLIDE

Add pin for OLED display VSS



life.augmented

SKIPPED SLIDE

I2C3 Configuration

Pinout & Configuration

Additional Softwares | Pinout

Options | Categories | A->Z

- System Core
- Analog
- Timers
- Connectivity
  - I2C1
  - I2C3
  - IRTIM
  - LPUART1
  - QUADSPI
  - RF
  - SPI1
  - SPI2
  - USART1
  - USB
- Multimedia
- Security
- Computing
- Middleware

I2C3 Mode and Configuration

Mode

I2C

Configuration

Reset Configuration

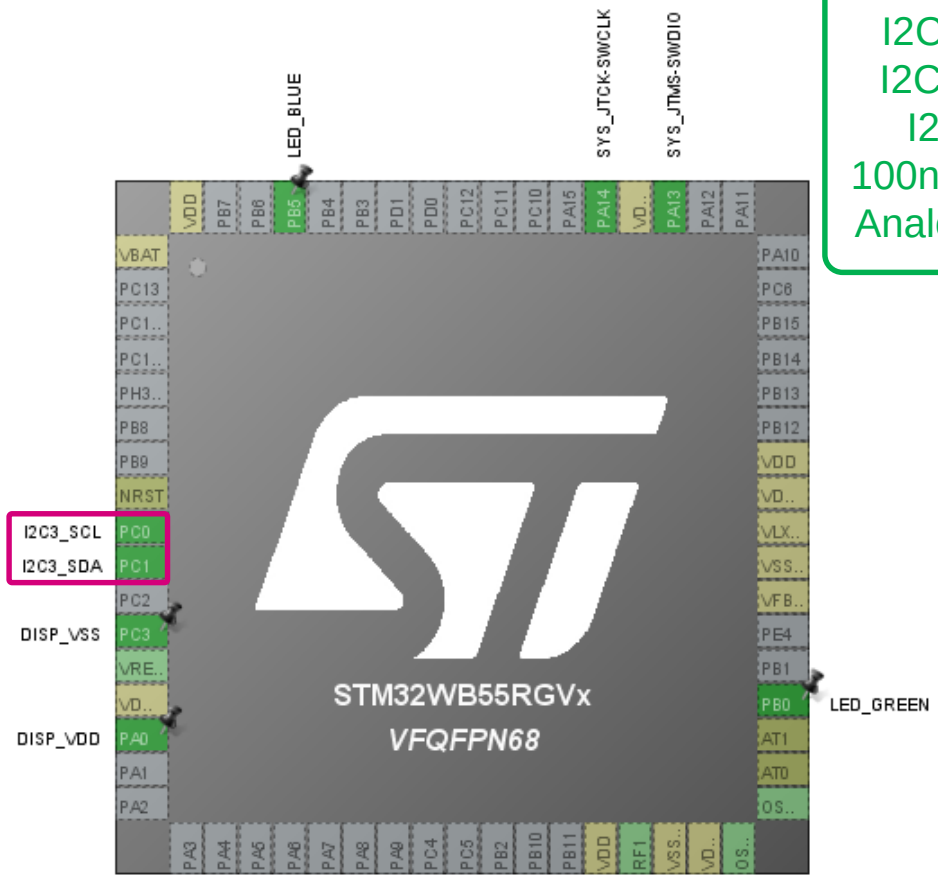
DMA Settings | GPIO Settings | User Constants | NVIC Settings | Parameter Settings

Configure the below parameters :

Search (Crtl+F)

Timing configuration	
Custom Timing	Disabled
I2C Speed Mode	Fast Mode
I2C Speed Freq...	400
Rise Time (ns)	100
Fall Time (ns)	100
Coefficient of Di...	0
Analog Filter	Enabled
Timing	0x00000003
Slave Features	
Clock No Stretc...	Disabled
General Call Ad...	Disabled
Primary Addres...	7-bit
Dual Address A...	Disabled
Primary slave ad...	0

Pinout view | System view



I2C3 in I2C mode
I2C3_SCL @PC0
I2C3_SDA @PC1
I2C Fast Mode
100ns Rise/Fall Time
Analog filter enabled

life.augmented

SKIPPED SLIDE

Add USART1

Pinout & Configuration

Additional Softwares | Pinout

Options | Categories | A->Z

- System Core
- Analog
- Timers
- Connectivity
  - I2C1
  - I2C3
  - IRTIM
  - LPUART1
  - QUADSPI
  - RF
  - SPI1
  - SPI2
  - USART1
  - USB
- Multimedia
- Security
- Computing
- Middleware

USART1 Mode and Configuration

Mode

Mode: Asynchronous

Hardware Flow Control (RS232): Disable

Hardware Flow Control (RS485)

Slave Select(NSS) Management: Disable

Configuration

Reset Configuration

DMA Settings | GPIO Settings | User Constants | NVIC Settings | Parameter Settings

Configure the below parameters :

Search (Crtl+F)

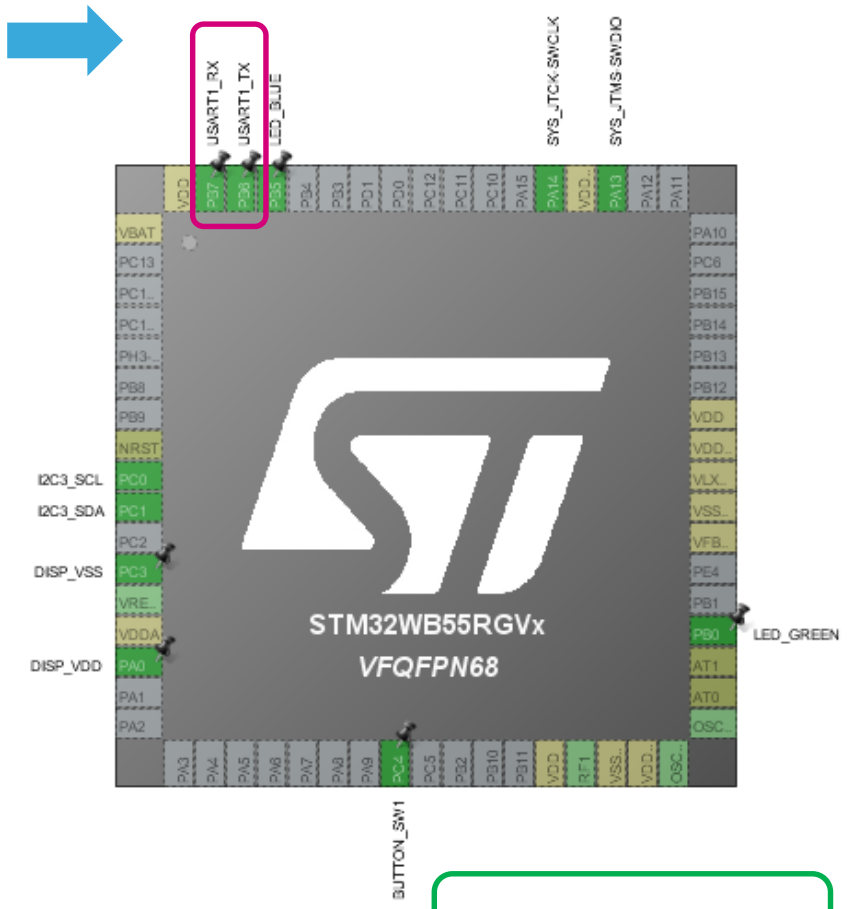
Basic Parameters

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters

Advanced Features

Pinout view | System view

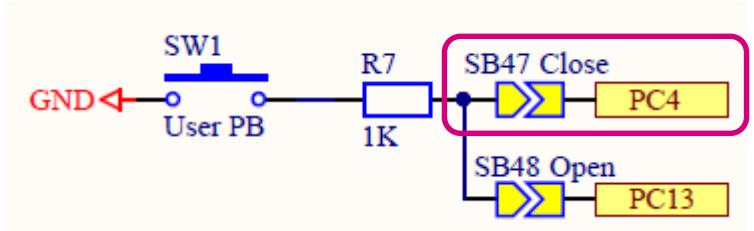


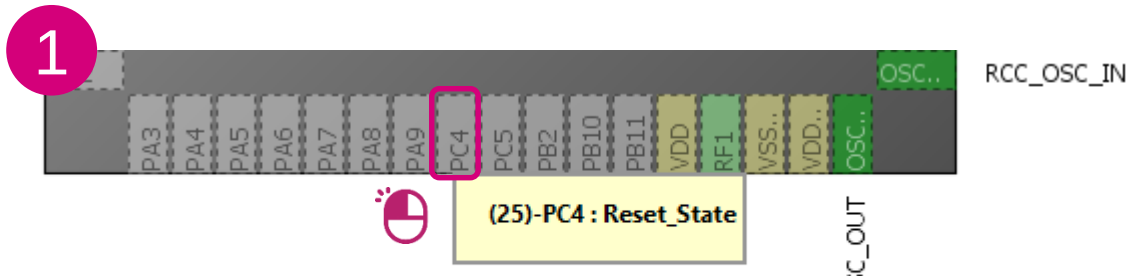
115200 bits/s
8bits (no parity)

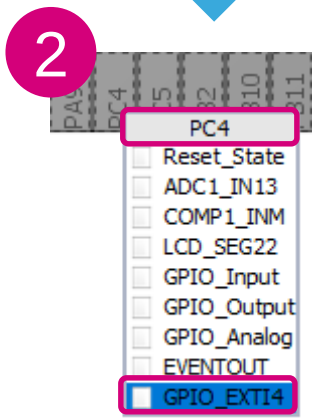
life.augmented

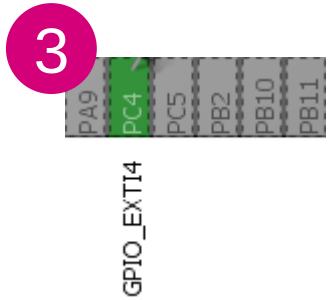
SKIPPED SLIDE

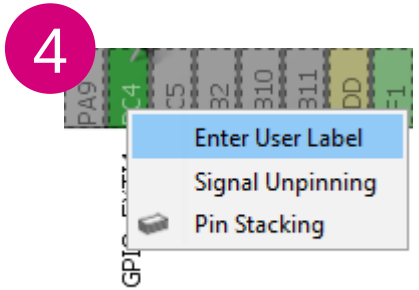
Add the SW1 Button EXTI input

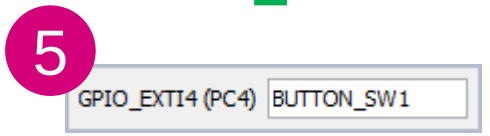


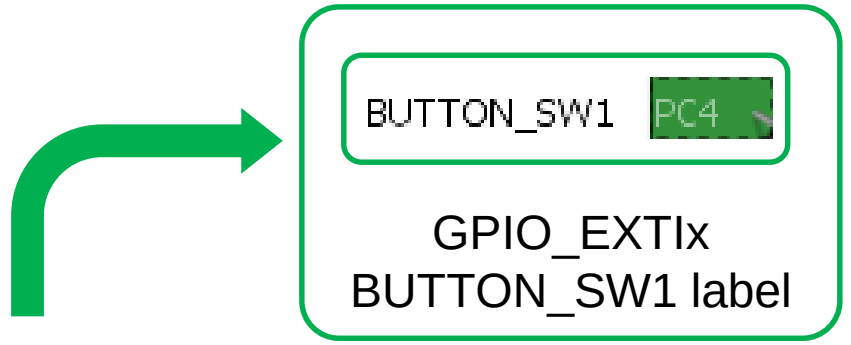








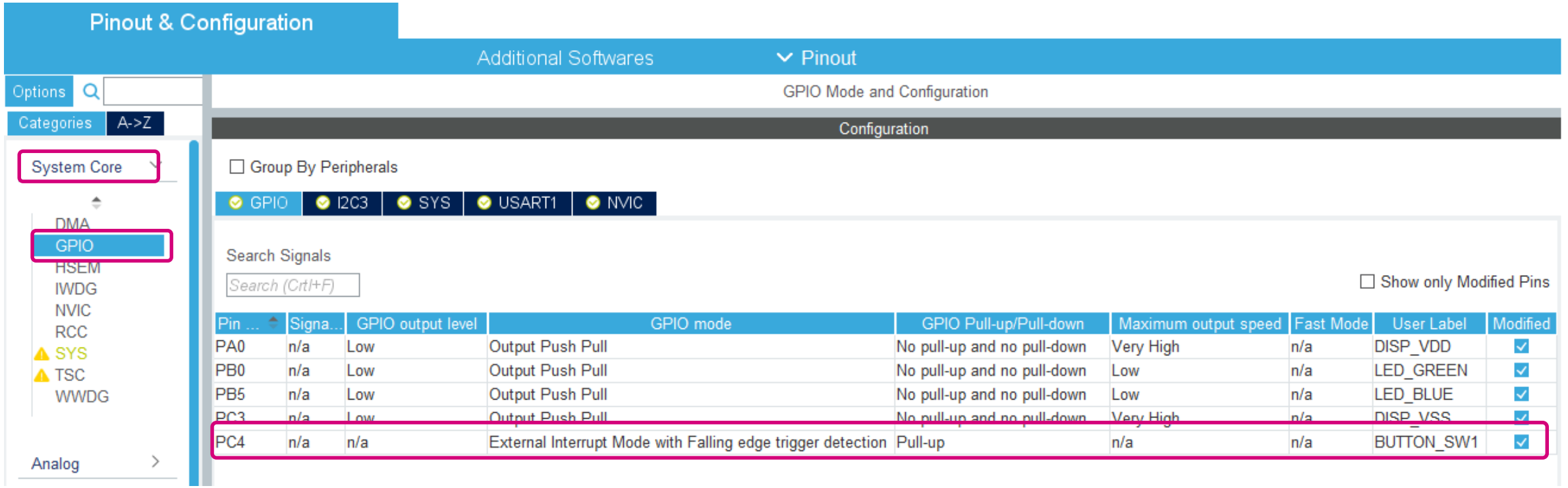




life.augmented

SKIPPED SLIDE

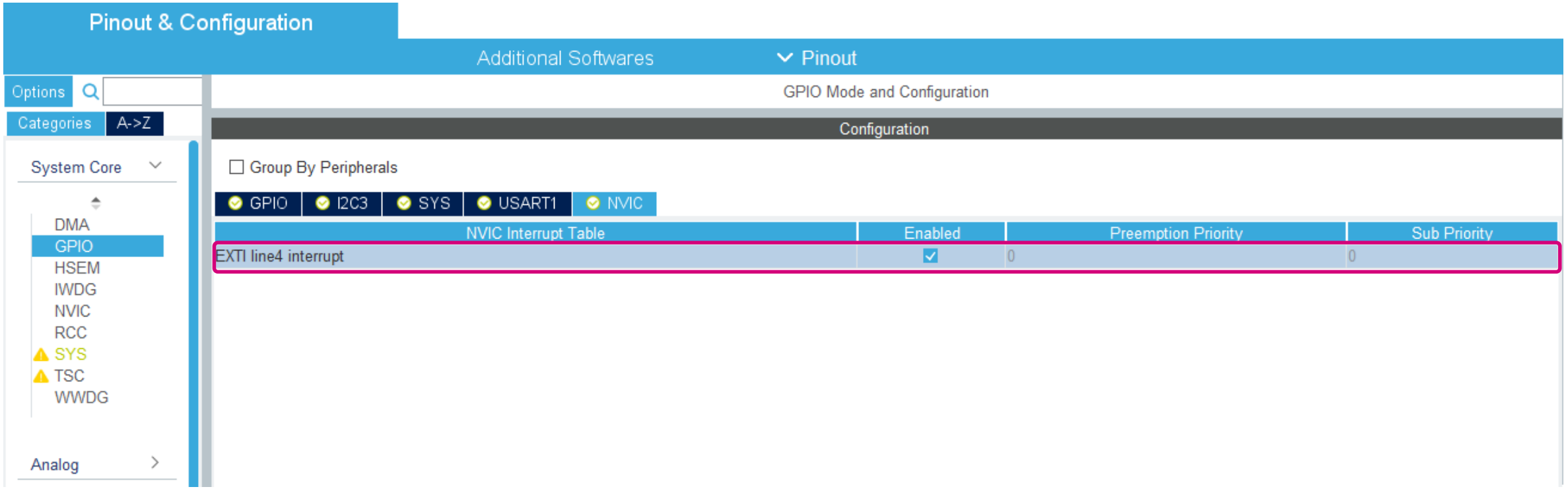
Configure Button SW1 EXTI input pin



System Core → GPIO → PC4
GPIO mode → External Interrupt Mode with Falling edge trigger detection
GPIO Pull-Up/Pull-Down → Pull-Up enabled

SKIPPED SLIDE

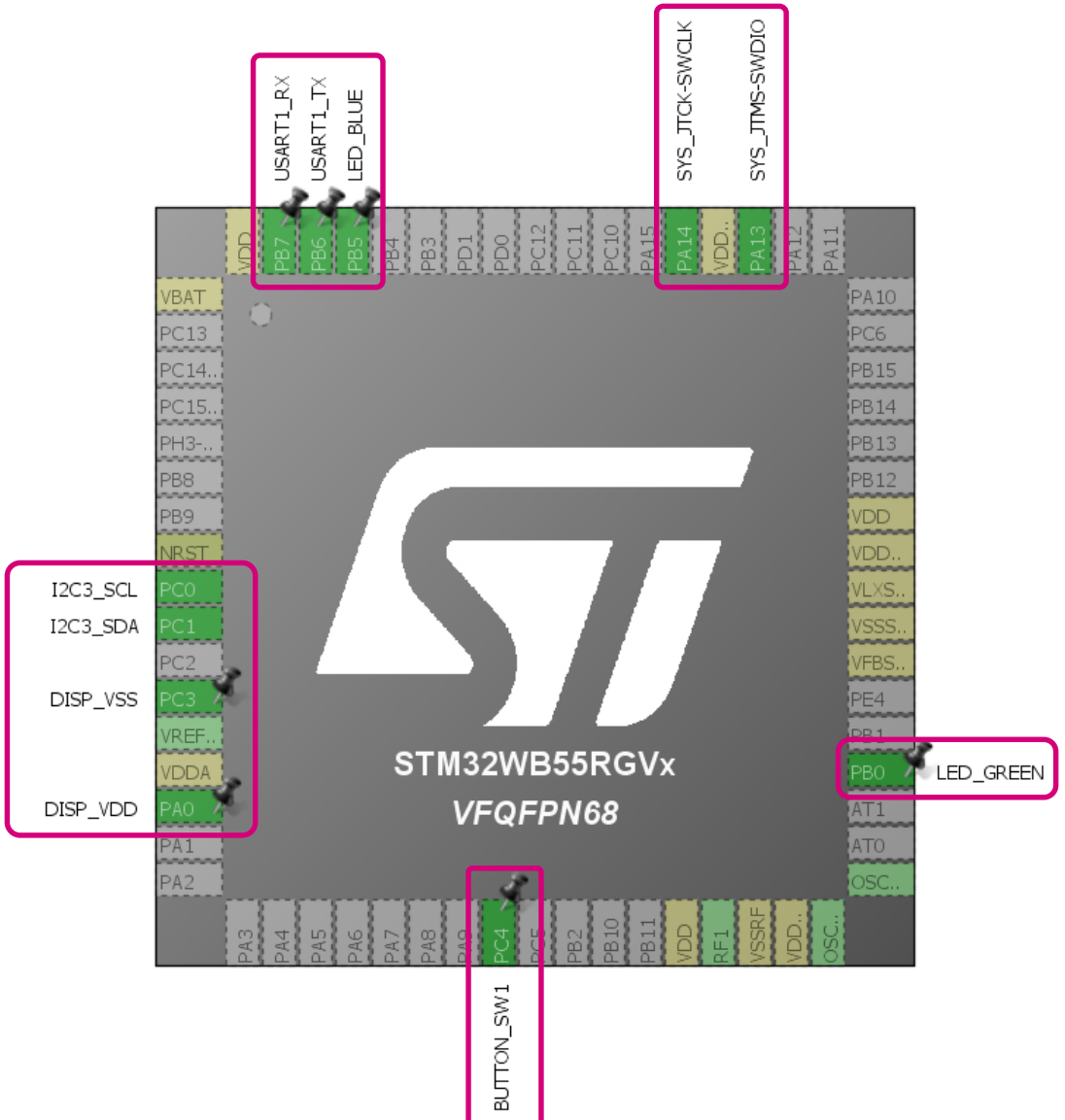
Enable EXTI line4 interrupt



System Core → GPIO → NVIC
EXTI line4 interrupt → Enabled

Let’s continue from this state

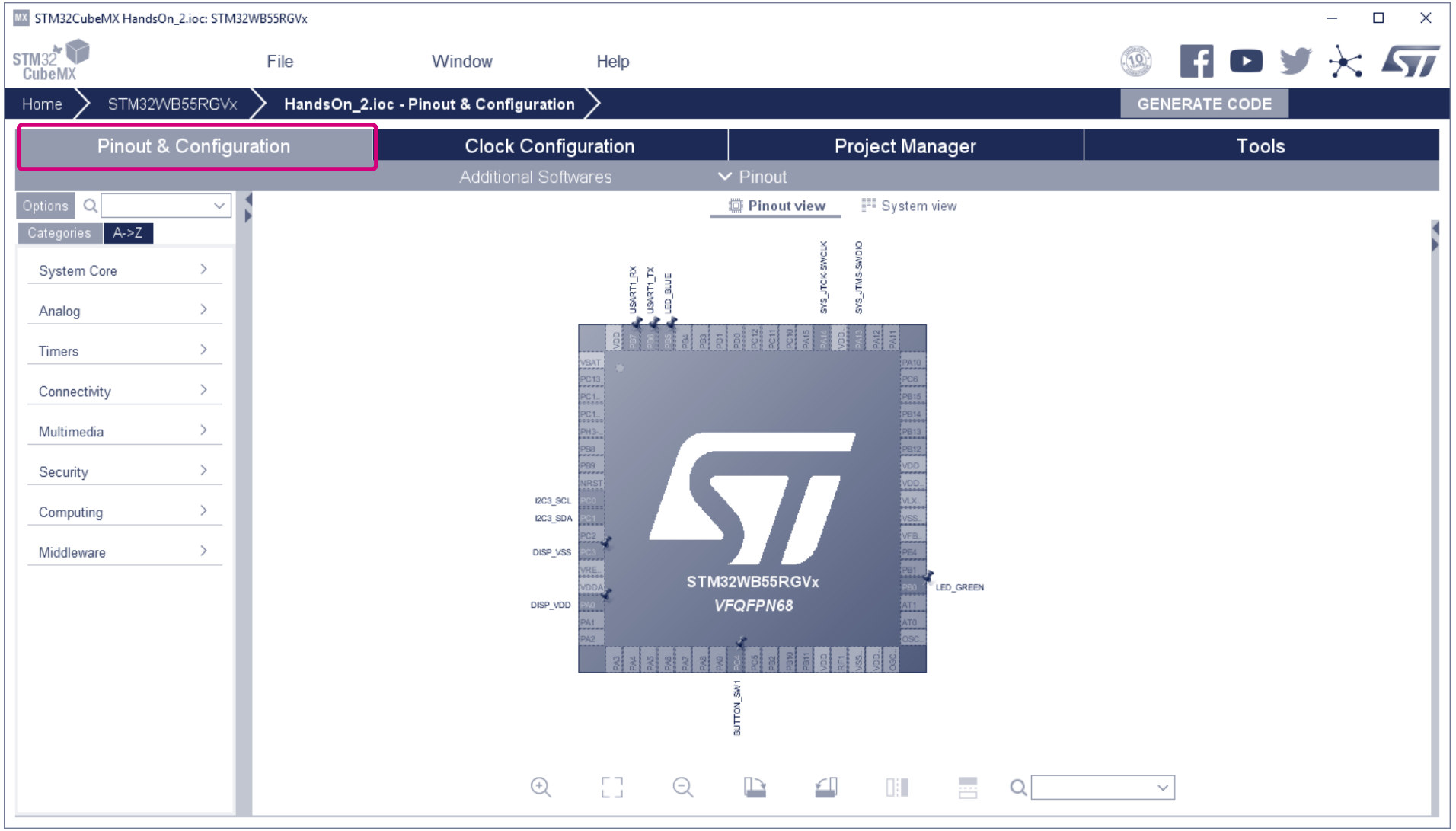
Move to this slide (Slide **10**)



I2C3_SCL @PC0
I2C3_SDA @PC1
DISP_VSS @PC3
DISP_VDD @PA0
LED_GREEN @PB0
LED_BLUE @PB5
SWD interface

life.augmented

Continue with Pinout Configuration



Add HSE crystal

Pinout & Configuration

Additional Softwares | Pinout

Options | Categories | A->Z

System Core
- DMA
- GPIO
- HSEM
- IWDG
- NVIC
- RCC
- SYS
- TSC
- WWDG

Analog

Timers

Connectivity

Multimedia

Security

Computing

Middleware

RCC Mode and Configuration

Mode

High Speed Clock (HSE): Crystal/Ceramic Resonator

Low Speed Clock (LSE): Disable

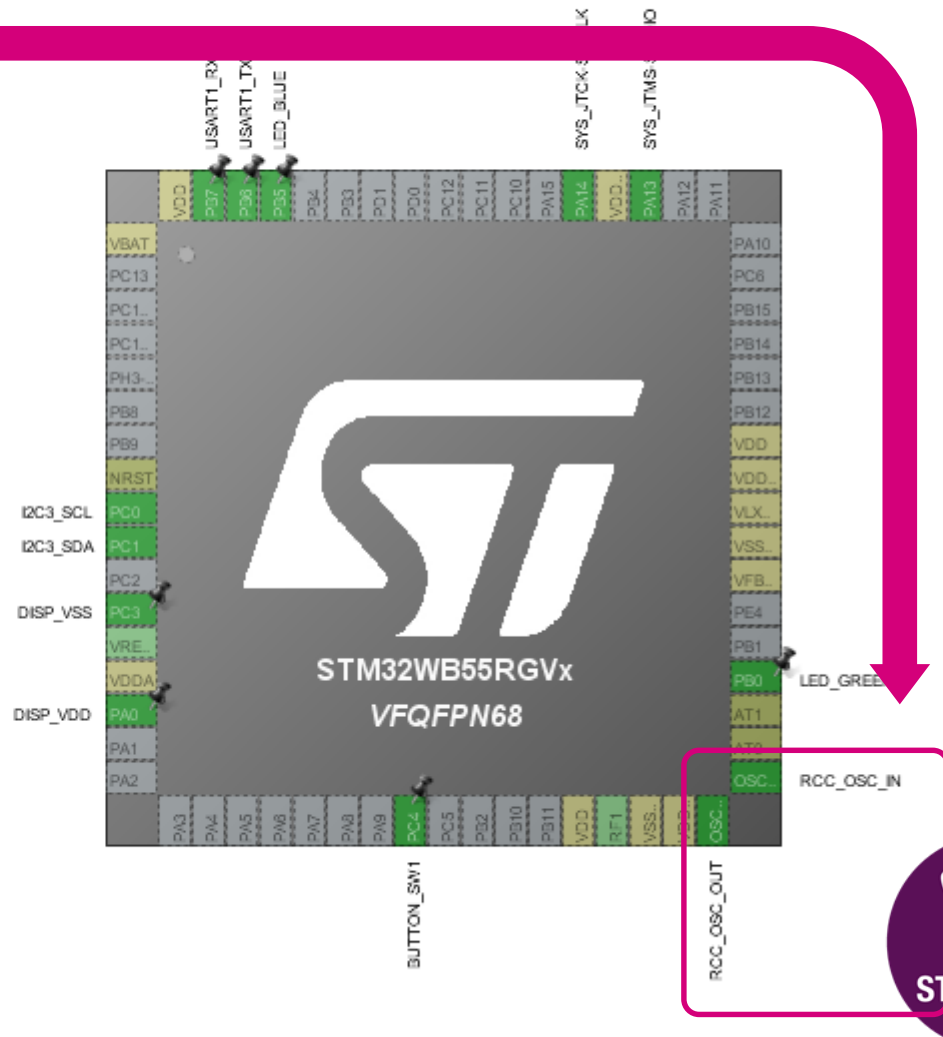
Master Clock Output

LSCO Clock Output

SAI1 Extern CLock

CRS SYNC: Disable

Pinout view | System view

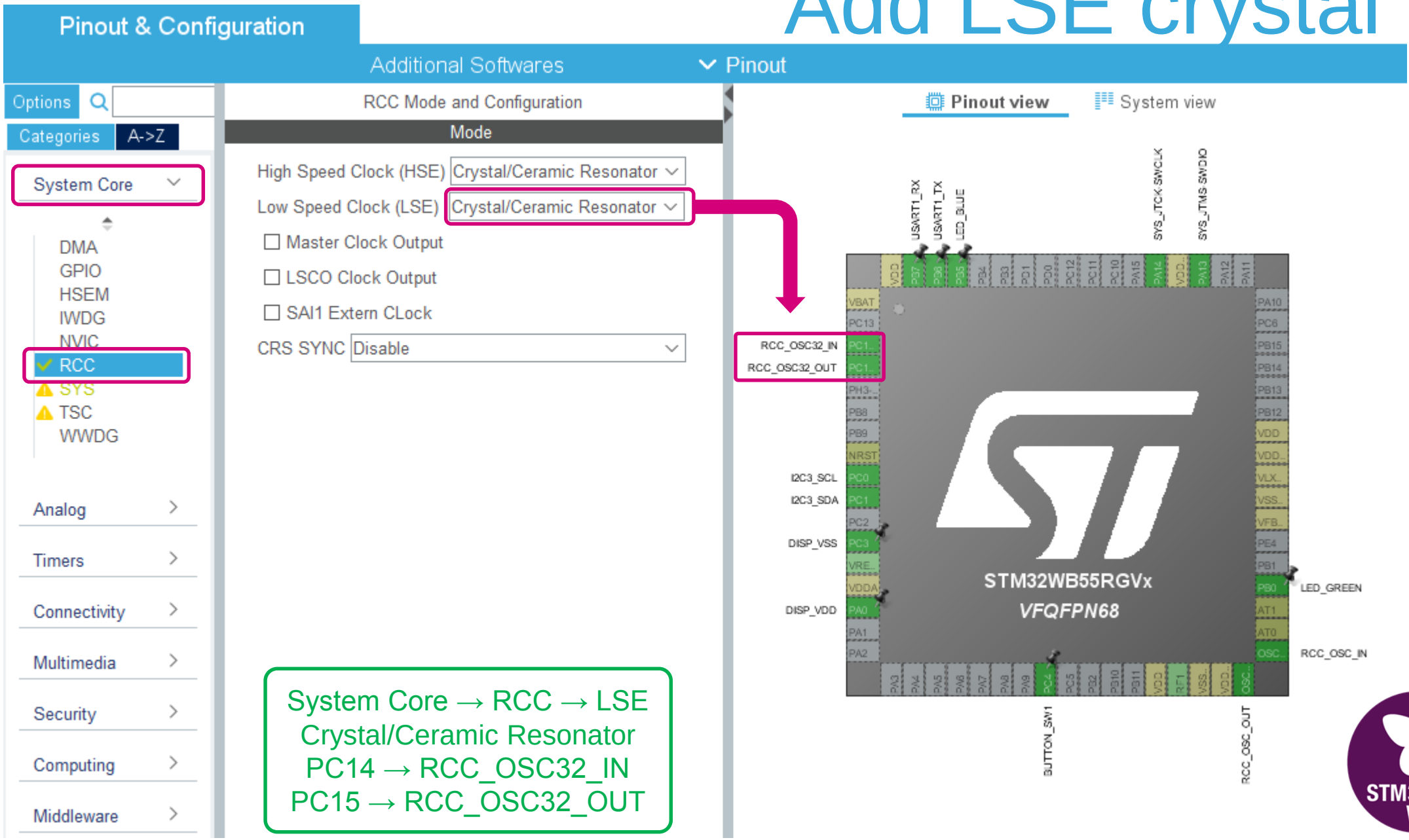


RCC → HSE
Crystal/Ceramic Resonator
RCC_OSC_OUT
RCC_OSC_IN

life.augmented

STM32 WB

Add LSE crystal

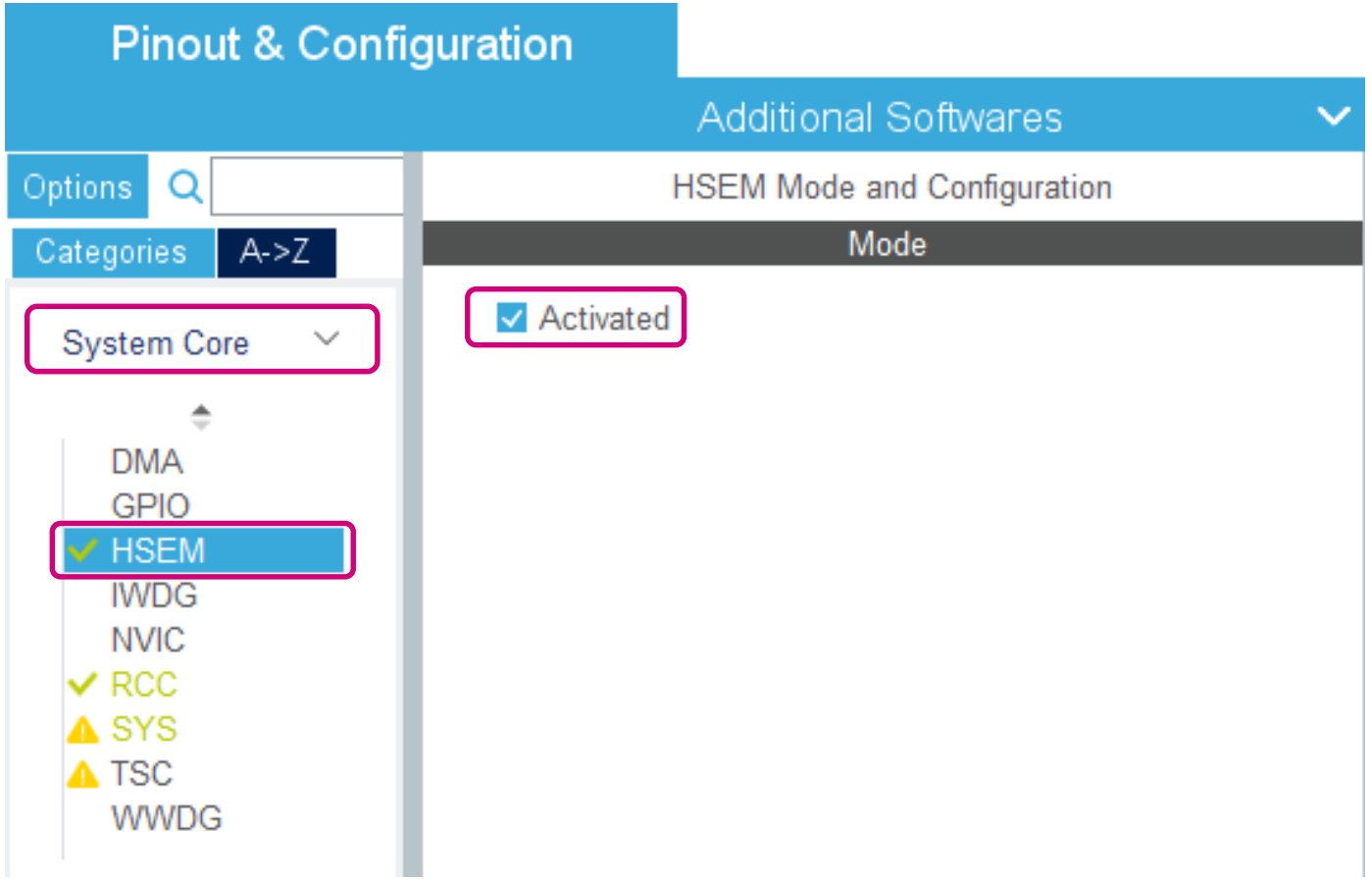


System Core → RCC → LSE
Crystal/Ceramic Resonator
PC14 → RCC_OSC32_IN
PC15 → RCC_OSC32_OUT

life.augmented

STM32 WB

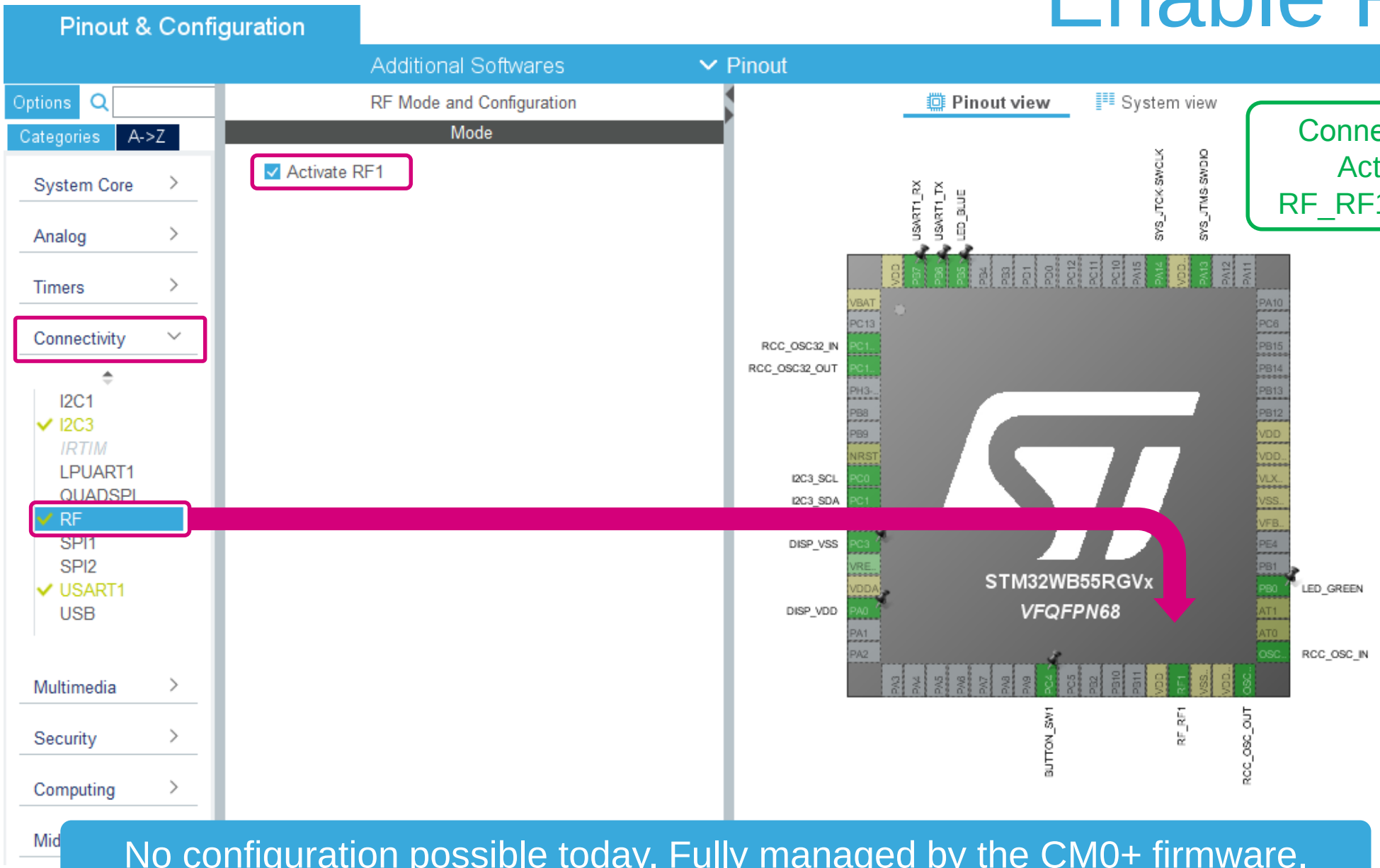
Enable Hardware Semaphores (HSEM)



System Core → HSEM Activated

No configuration possible today. Fully managed by the middleware. Just to keep in mind that it exists and is in use to manage access to resources shared by both CM0+ and CM4F.

Enable RF



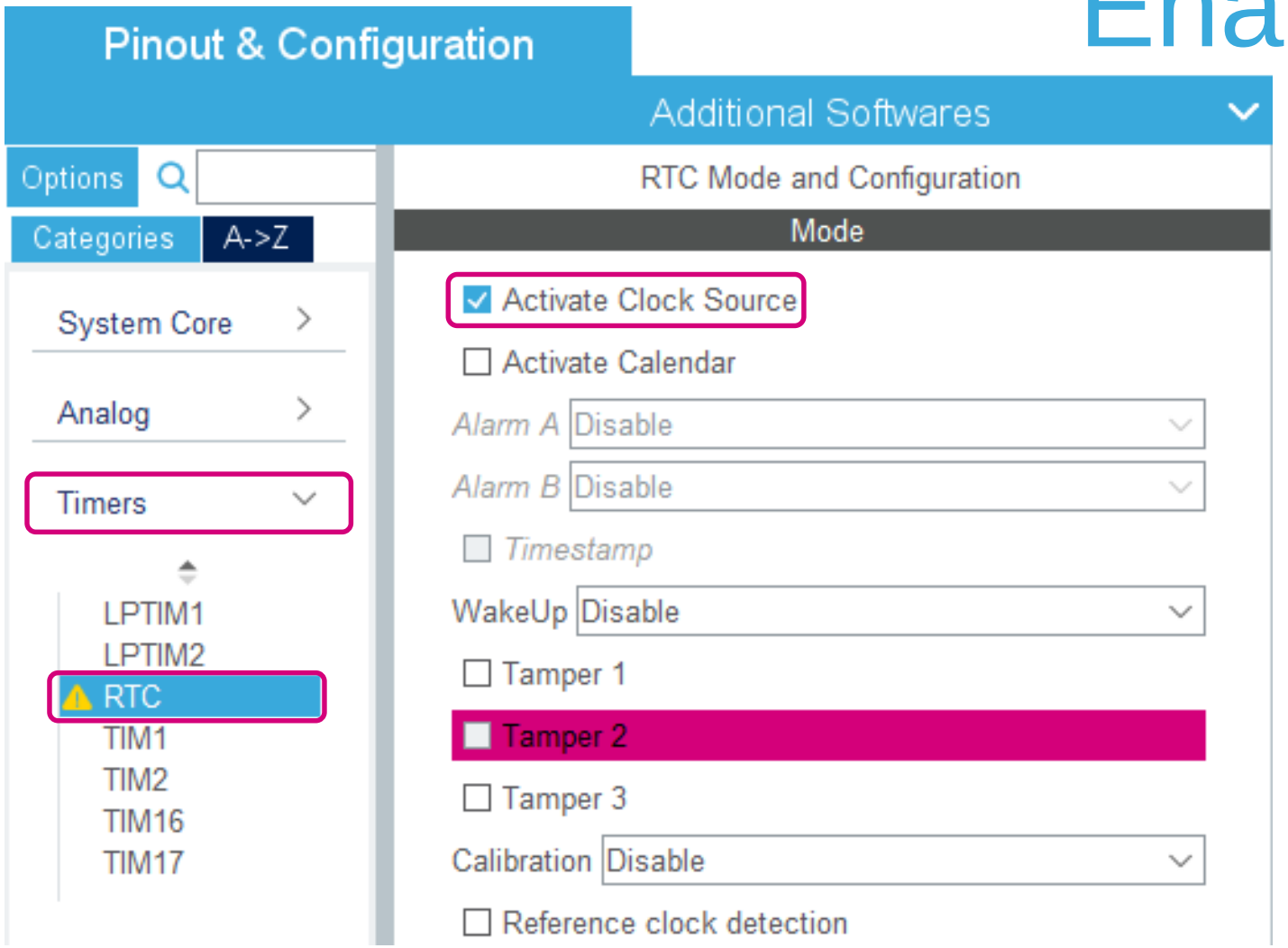
Connectivity → RF
Activate RF1
RF_RF1 pin assigned

No configuration possible today. Fully managed by the CM0+ firmware.

life.augmented

Enable RTC

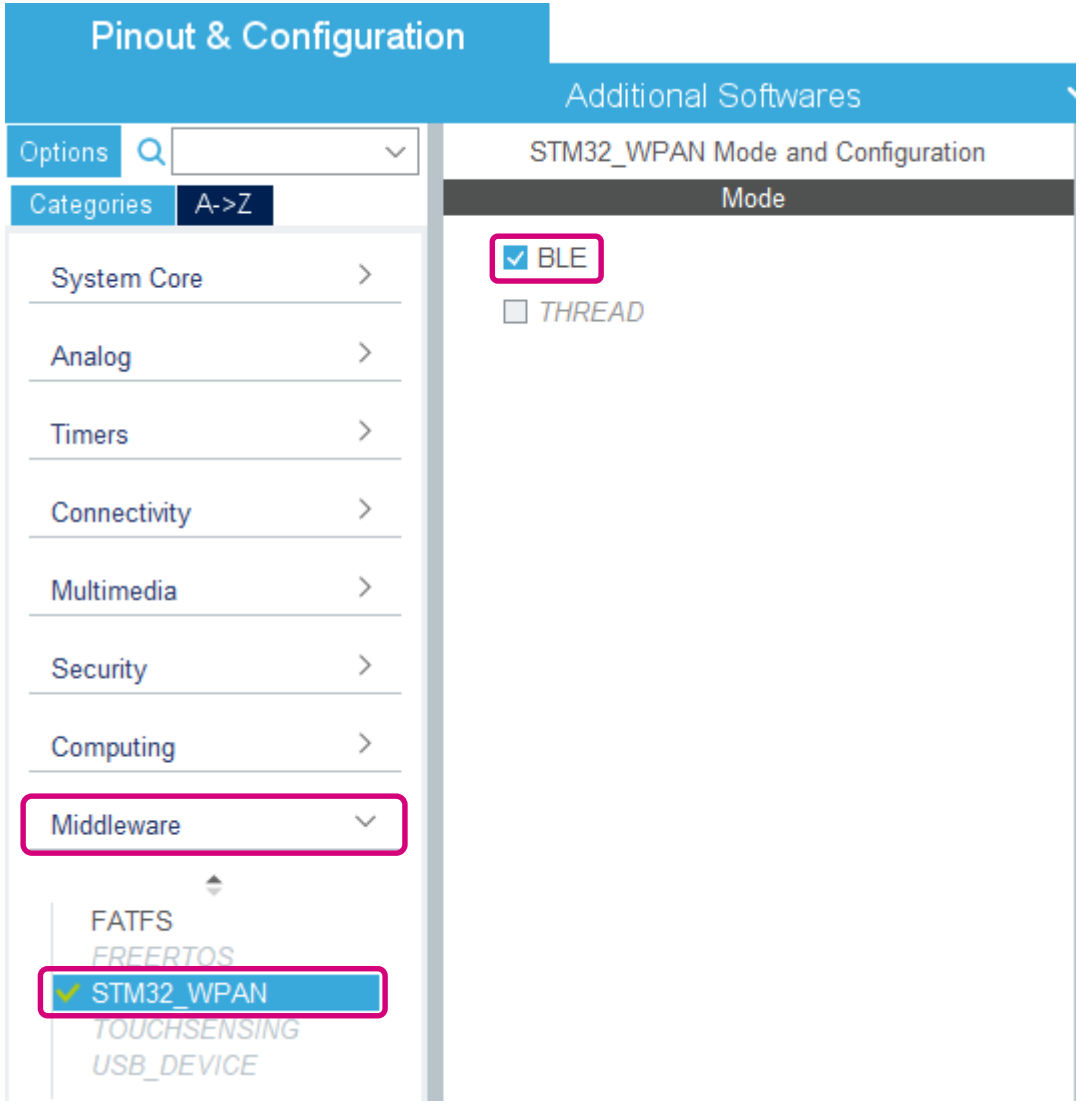
Timers → RTC
Activate Clock Source



RTC used by STM32_WPAN middleware just to provide some timebase for SW timers and Low-Power modes support.
Fully modifiable according to user application needs.

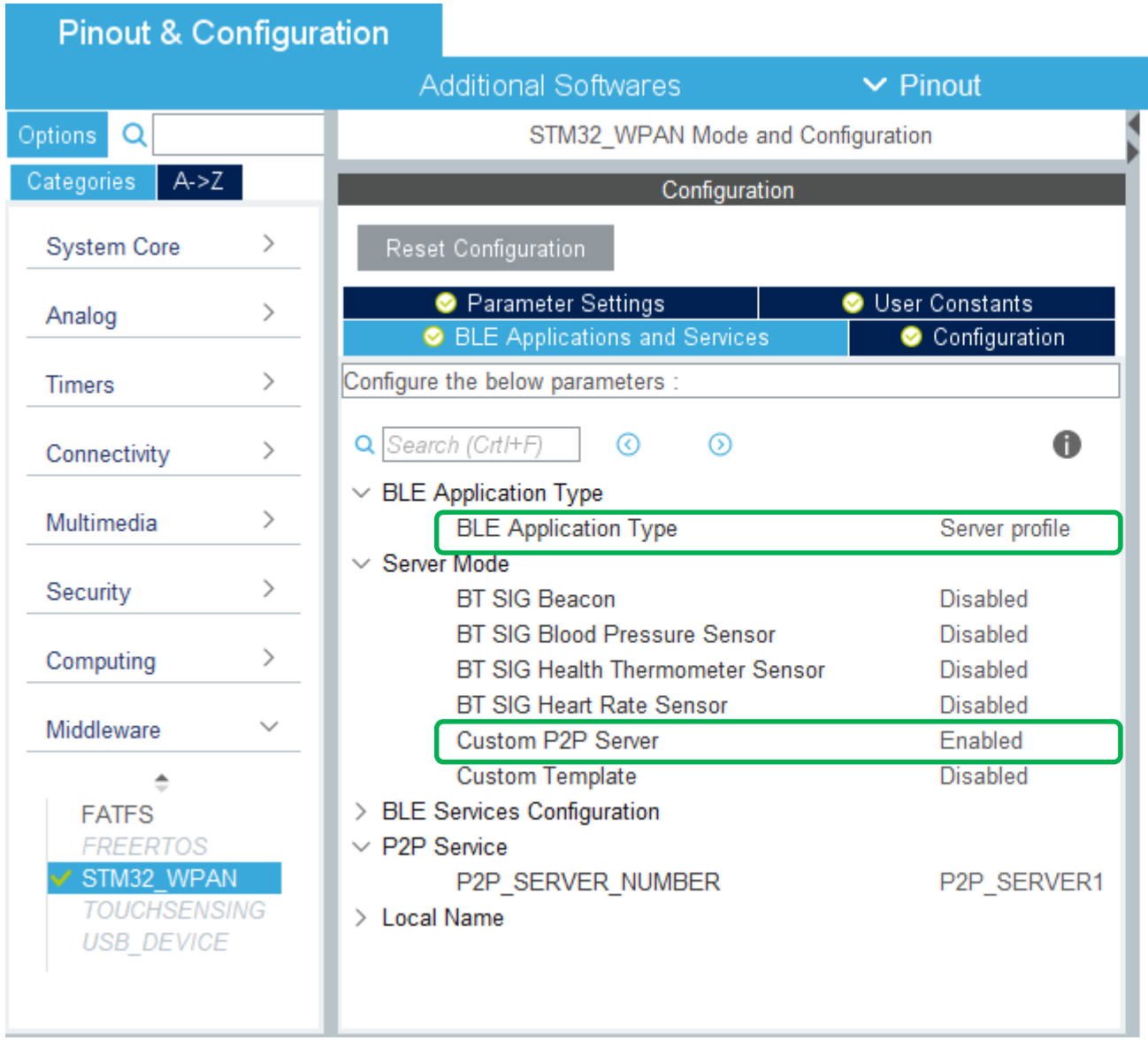
life.augmented

Enable STM32_WPAN BLE Middleware



Middleware → STM32_WPAN BLE

Configure STM32_WPAN BLE Middleware



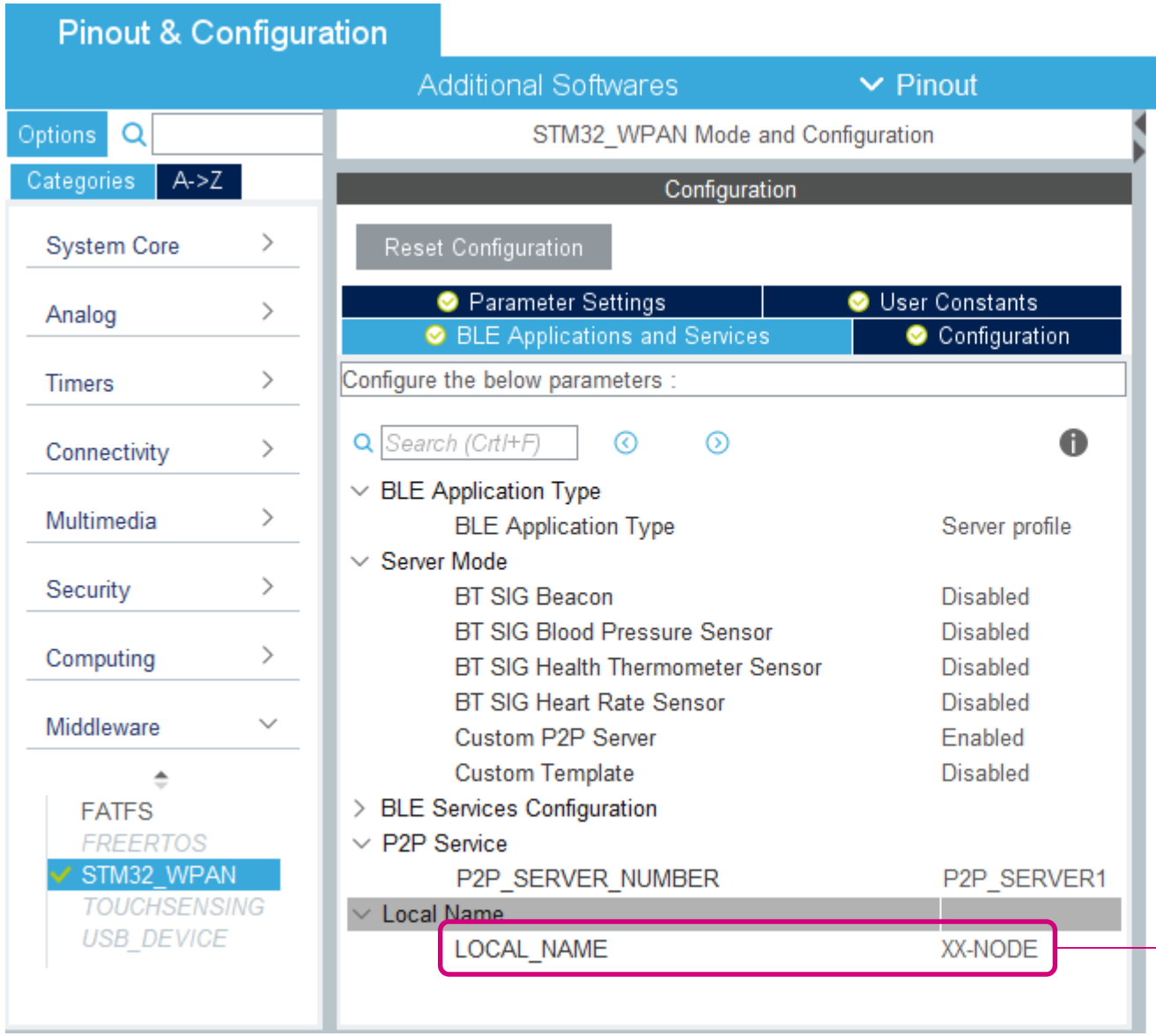
BLE Application Type → Server profile
Server Mode → Custom P2P Server Enabled

No need to change, already pre-configured.

Your very own happy number for today

01, 02,…. XX
To be used in your advertised complete local name.

Configure STM32_WPAN BLE Middleware



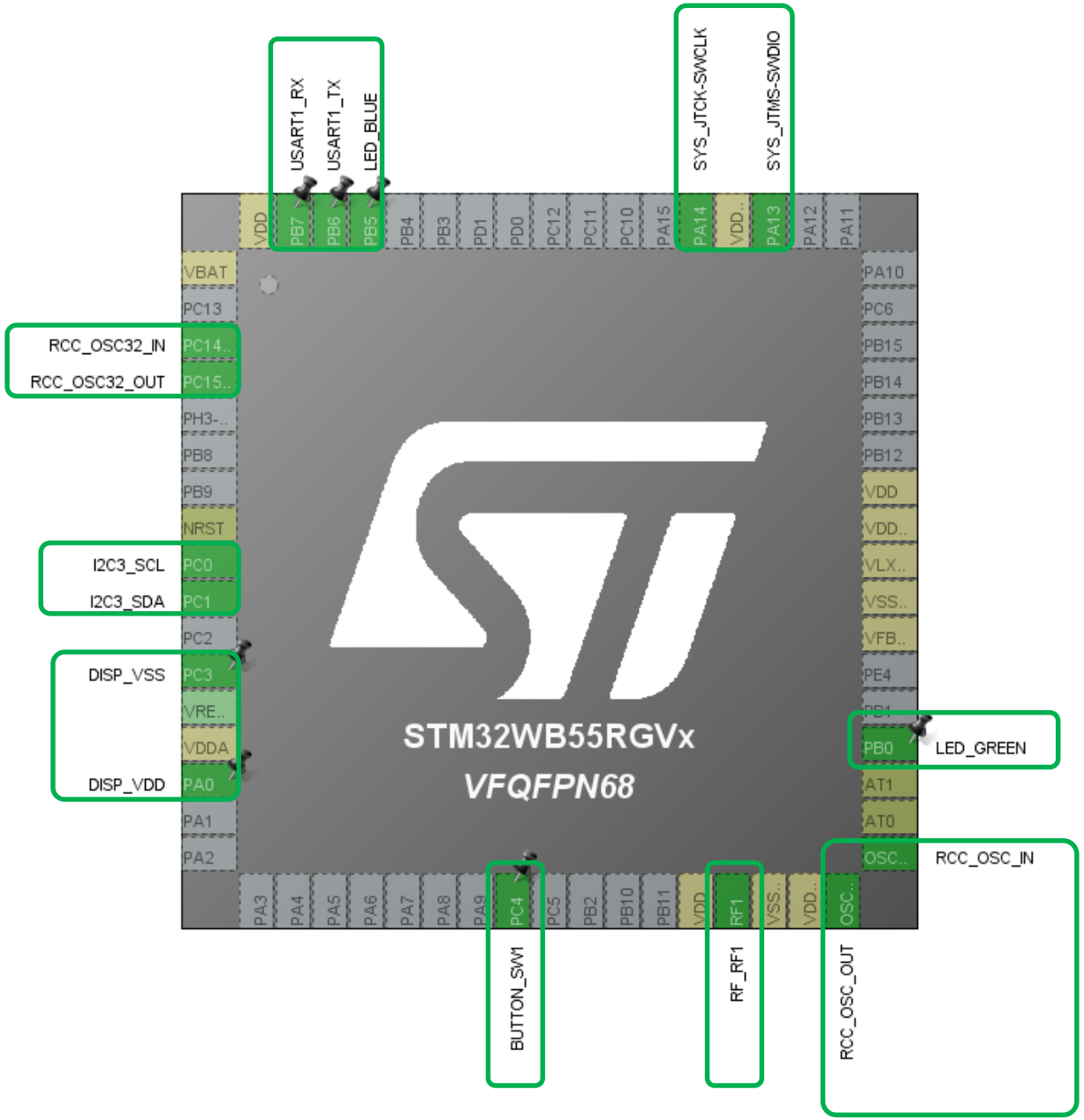
LOCAL_NAME changed according to your happy number

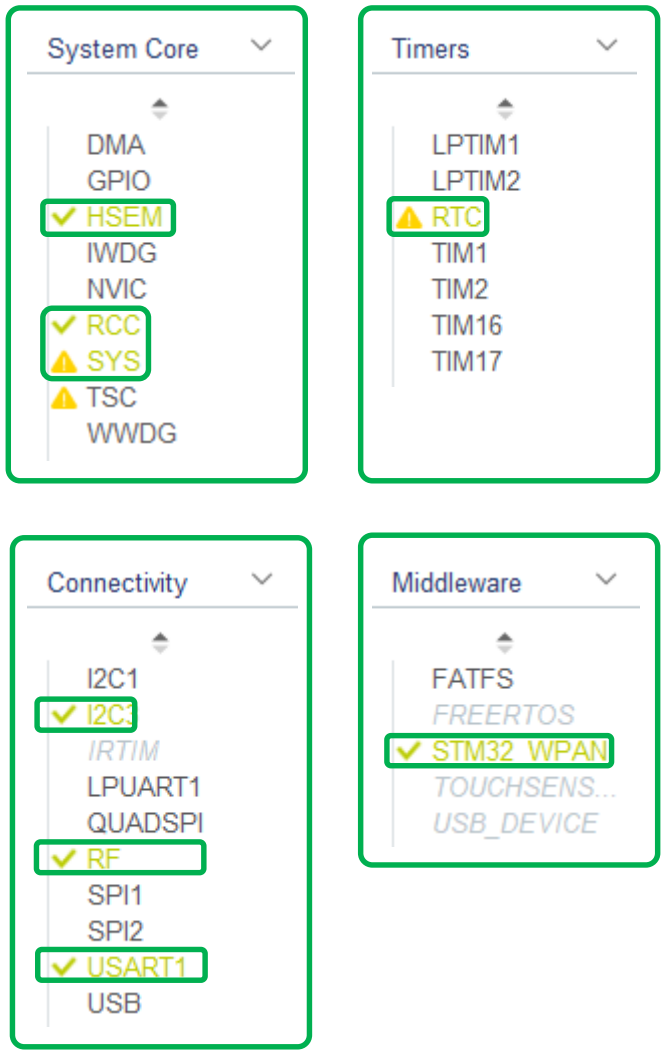
Max number of characters is set to 7 to be aligned with overall length of advertising data used by the generated code.

XX-NODE where **XX** is your happy number

Checkpoint

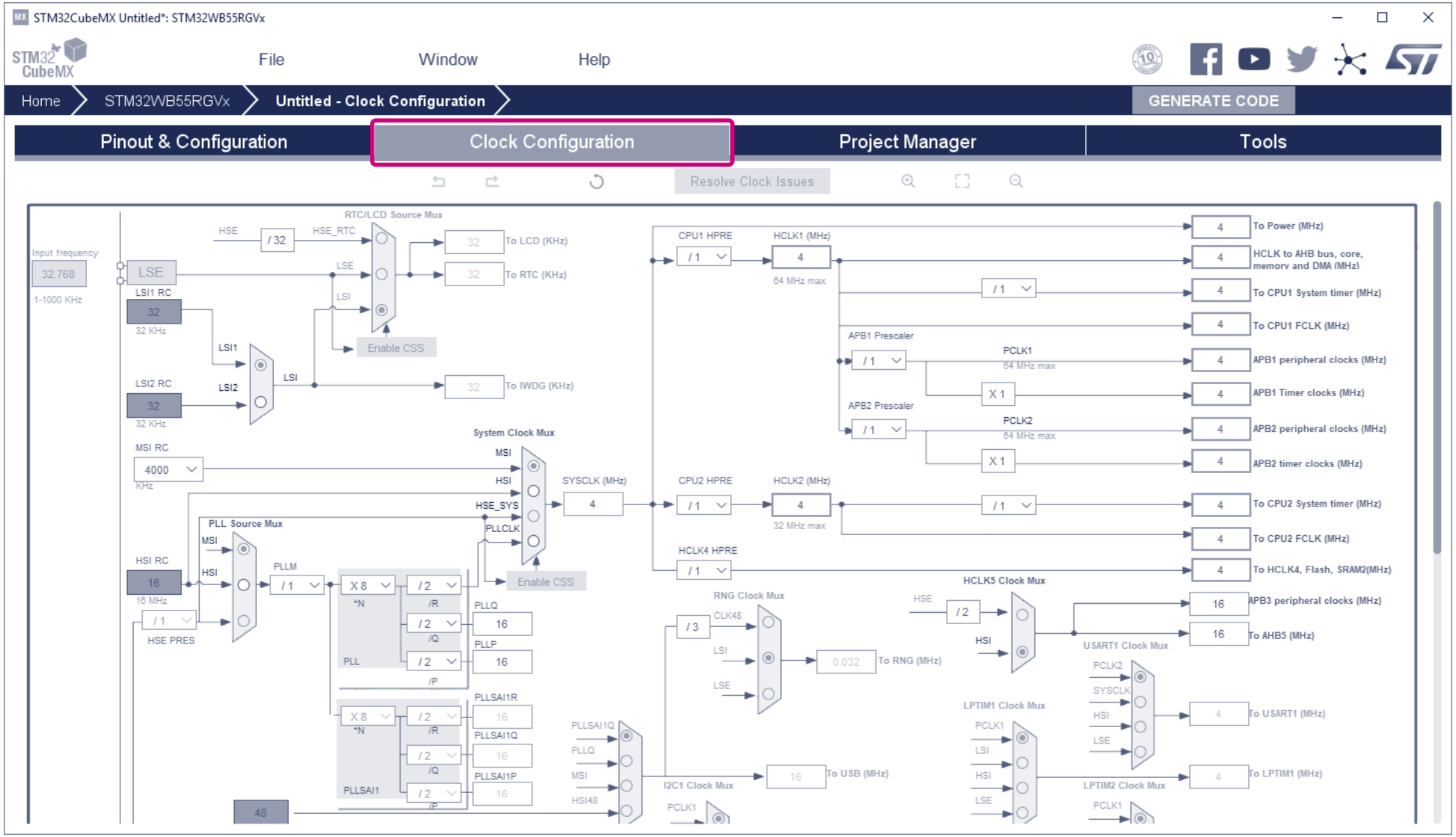
SWD @PA13/PA14
I2C3_SCL @PC0
I2C3_SDA @PC1
DISP_VSS @PC3
DISP_VDD @PA0
LED_GREEN @PB0
LED_BLUE @PB5
USART1_RX @PB7
USART1_TX @PB6
HSE
LSE
RTC Activated
HSEM Activated
RF Activated
STM32_WPAN BLE





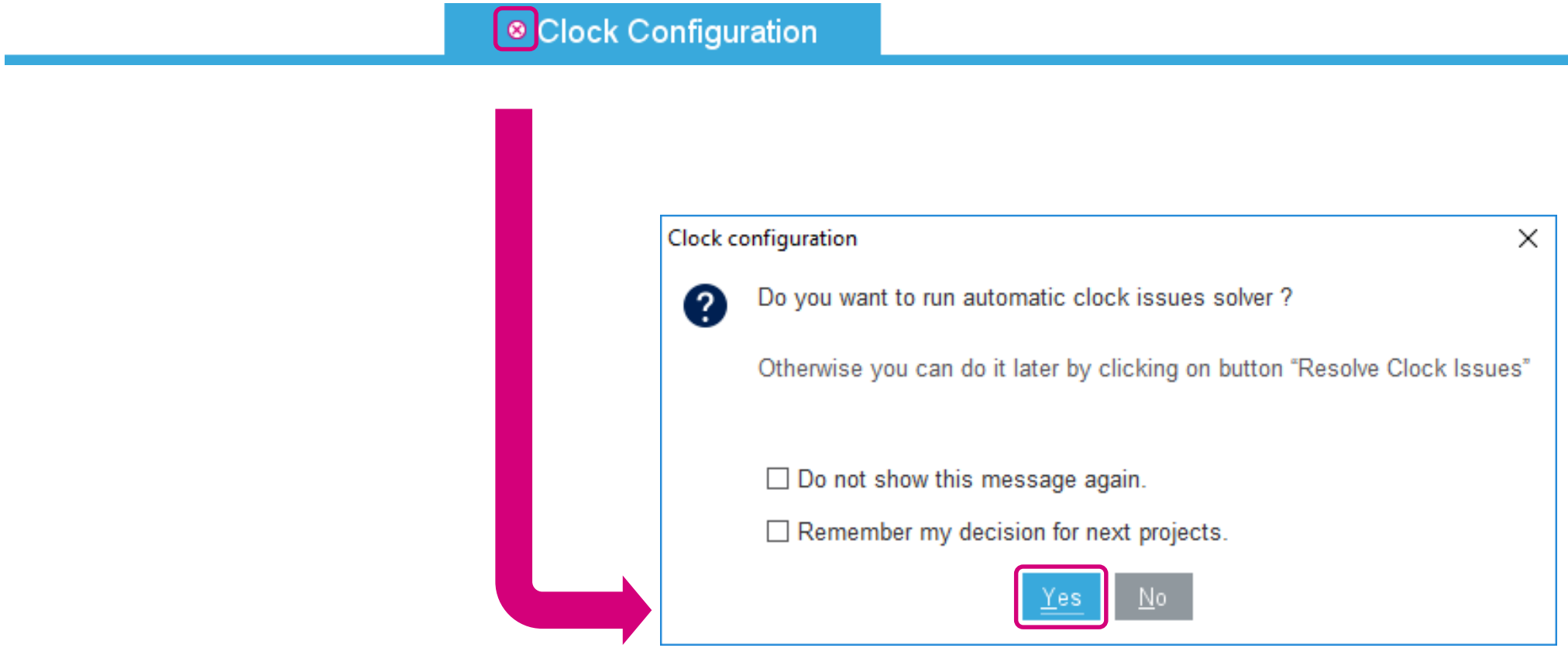
life.augmented

Switch to clock configuration



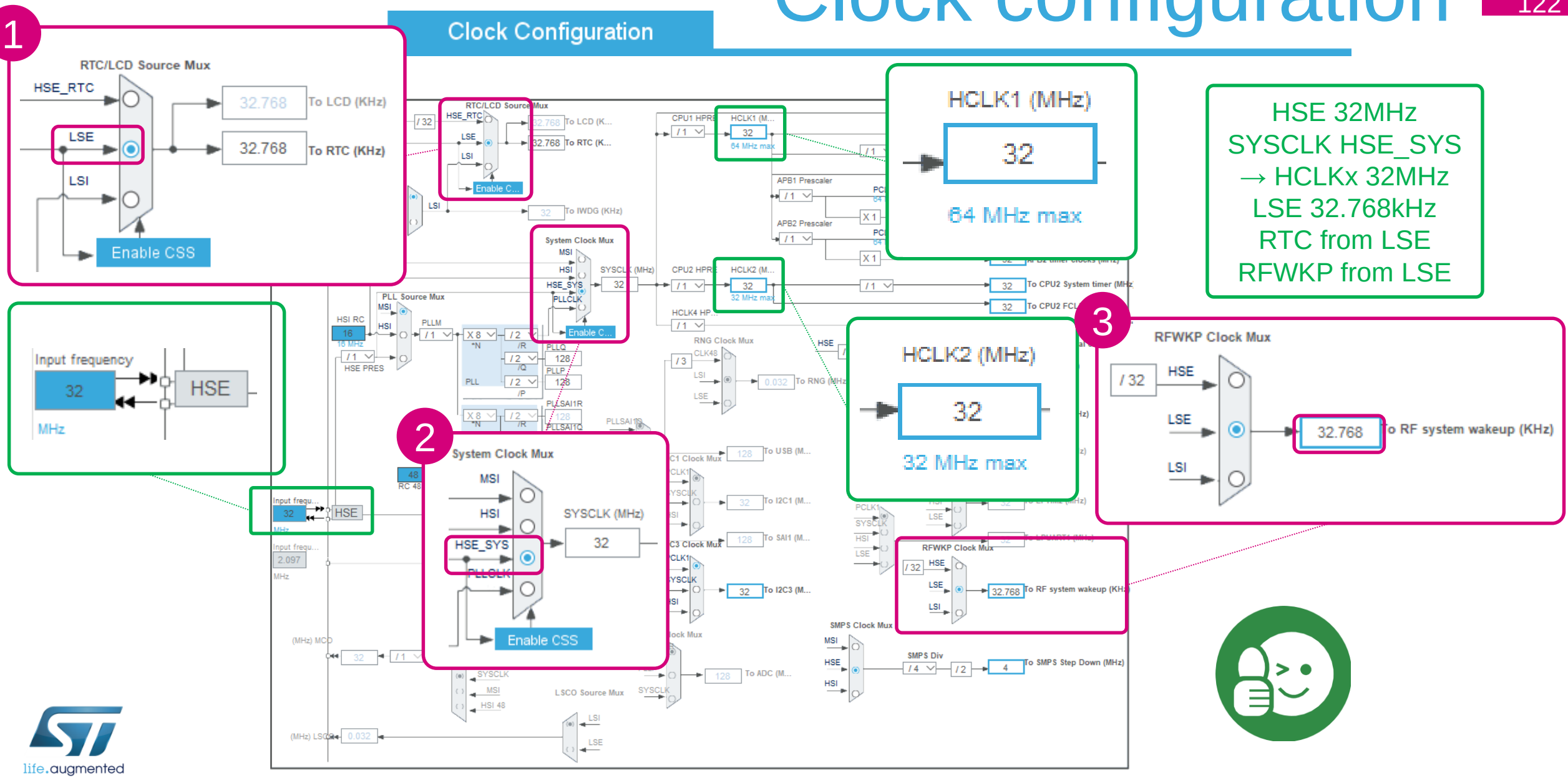
life.augmented

Clock issue solver

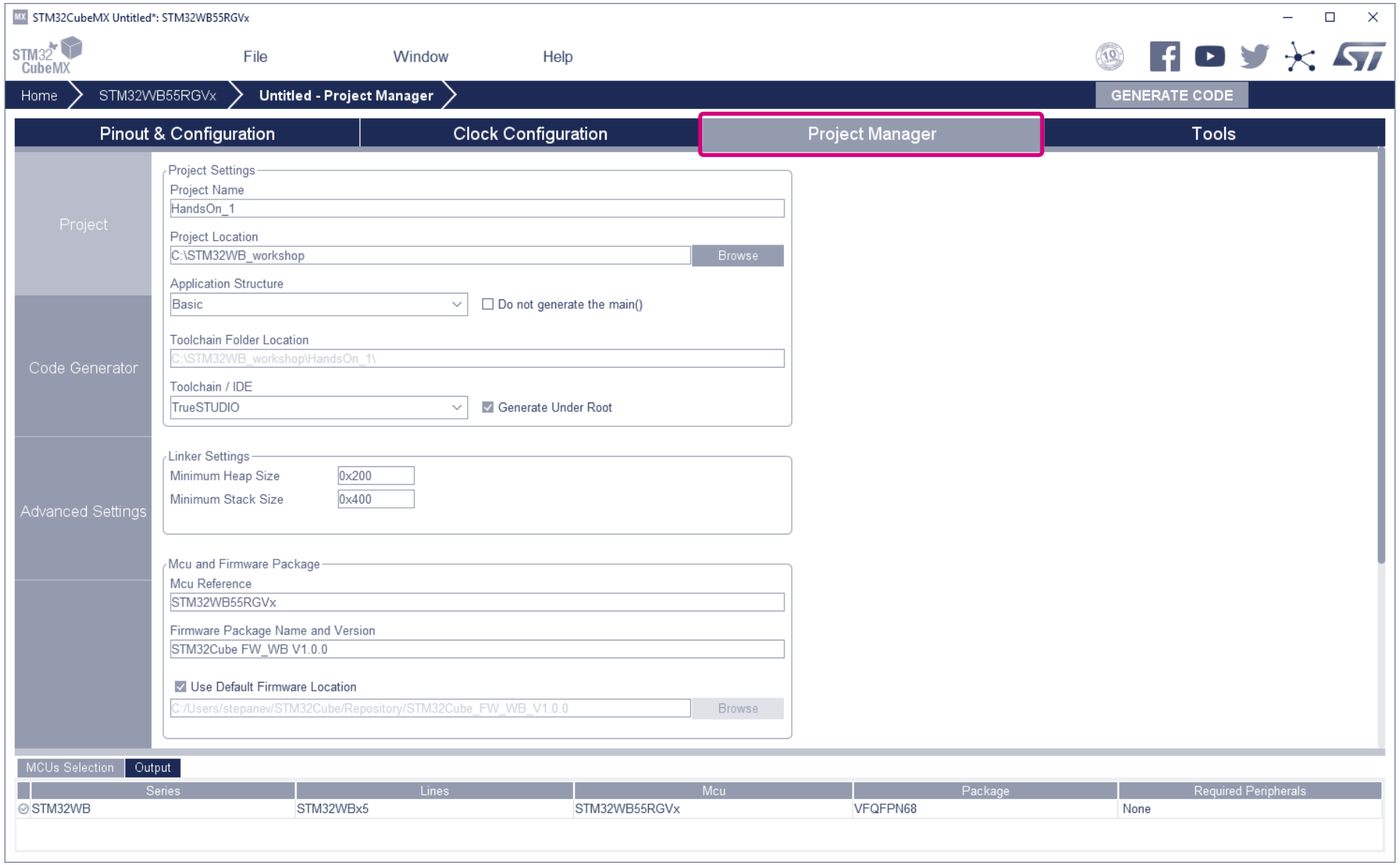


The clocks requirements for select peripherals (e.g. 32MHz for RF) are not fulfilled, click on Yes, when the Clock issue solver dialog pops up

Clock configuration



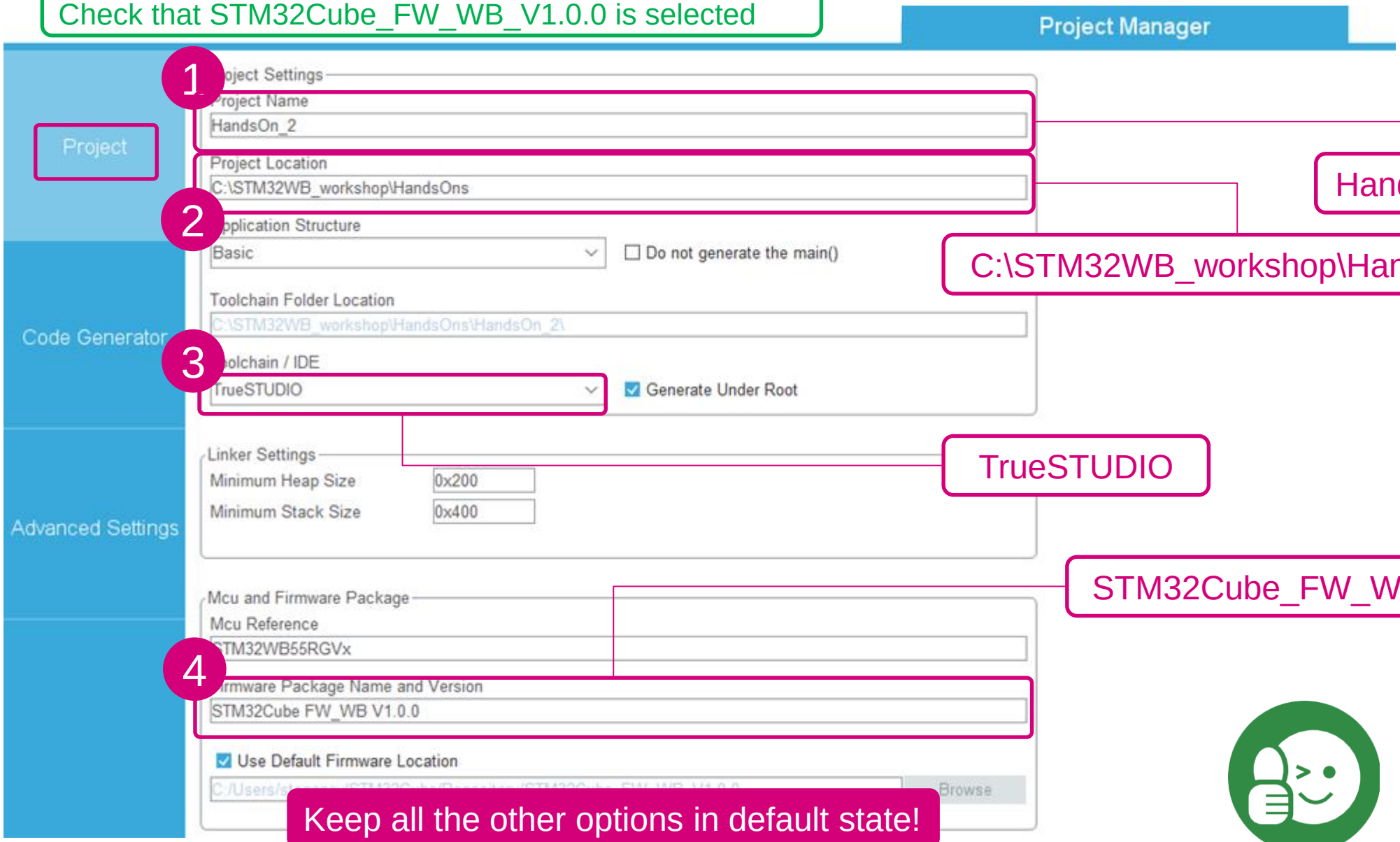
Finalize the project settings



life.augmented

Project Name → HandsOn_2
Project Location → C:\STM32WB_workshop\HandsOns\
IDE → TrueSTUDIO
Check that STM32Cube_FW_WB_V1.0.0 is selected

Project settings



HandsOn_2

C:\STM32WB_workshop\HandsOns

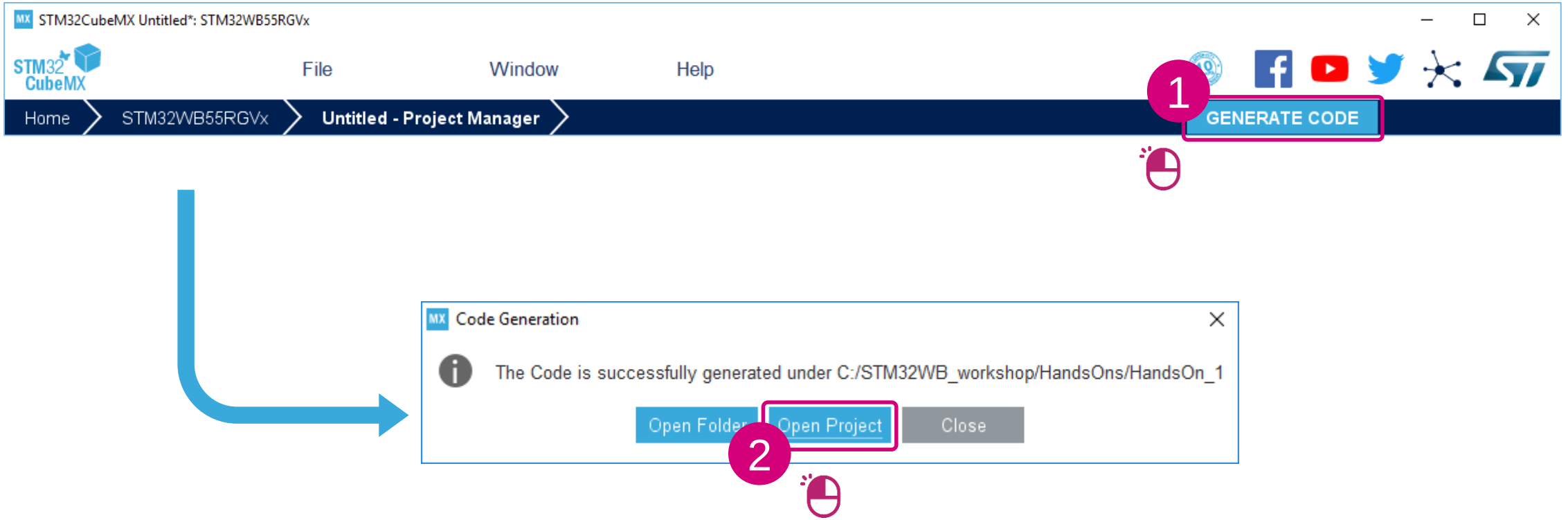
TrueSTUDIO

STM32Cube_FW_WB_V1.0.0

Keep all the other options in default state!

life.augmented

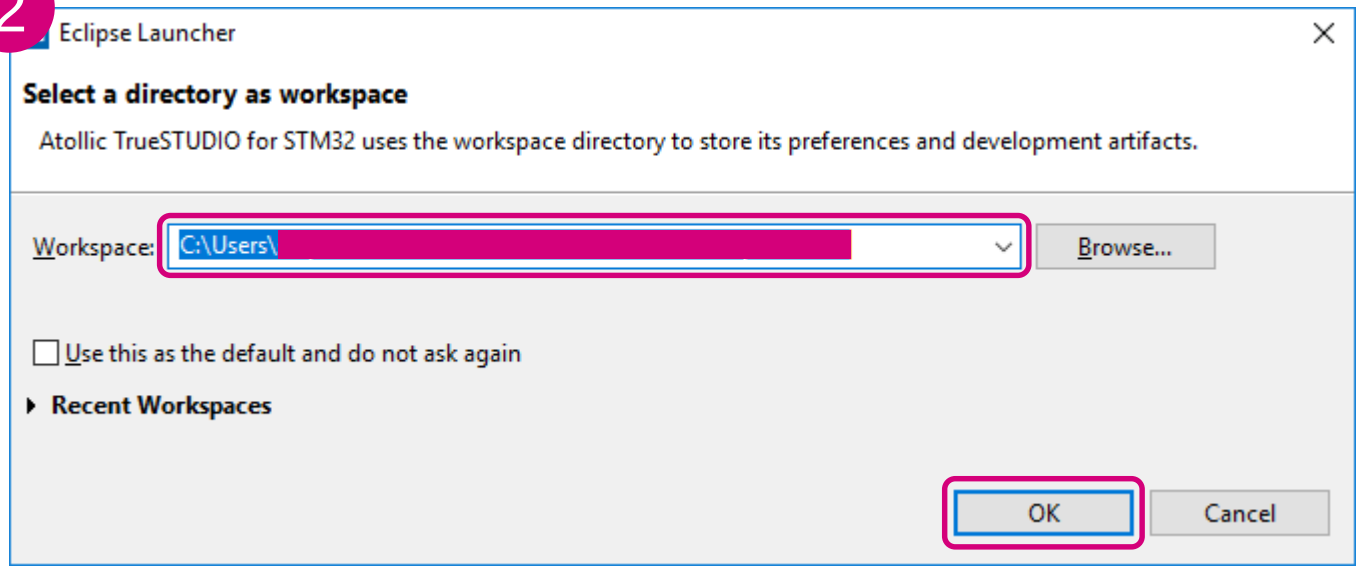
Generate the code



Atollic TrueStudio project opening

1

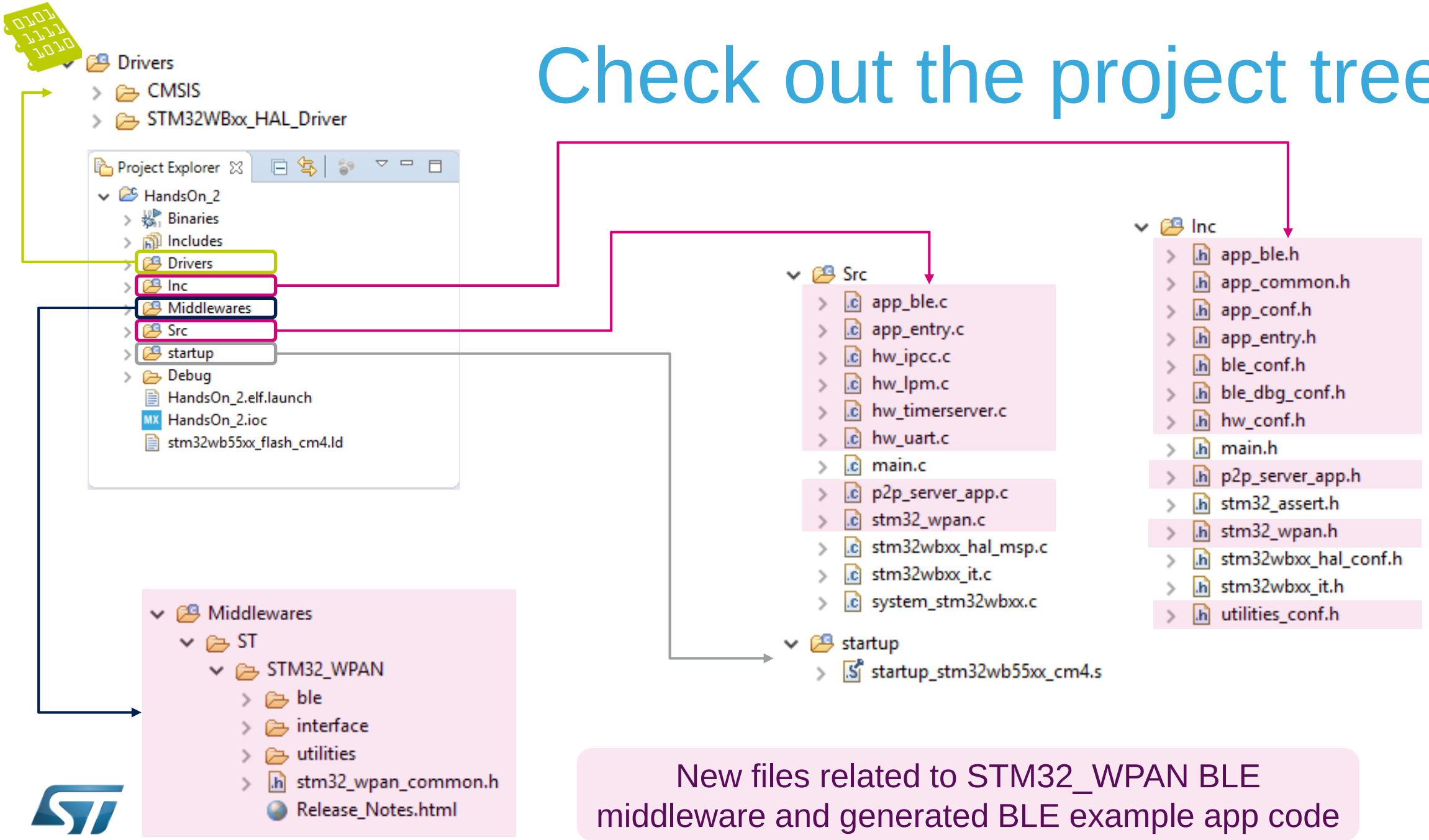
2



Select the workspace

life.augmented

Check out the project tree



New files related to STM32_WPAN BLE middleware and generated BLE example app code

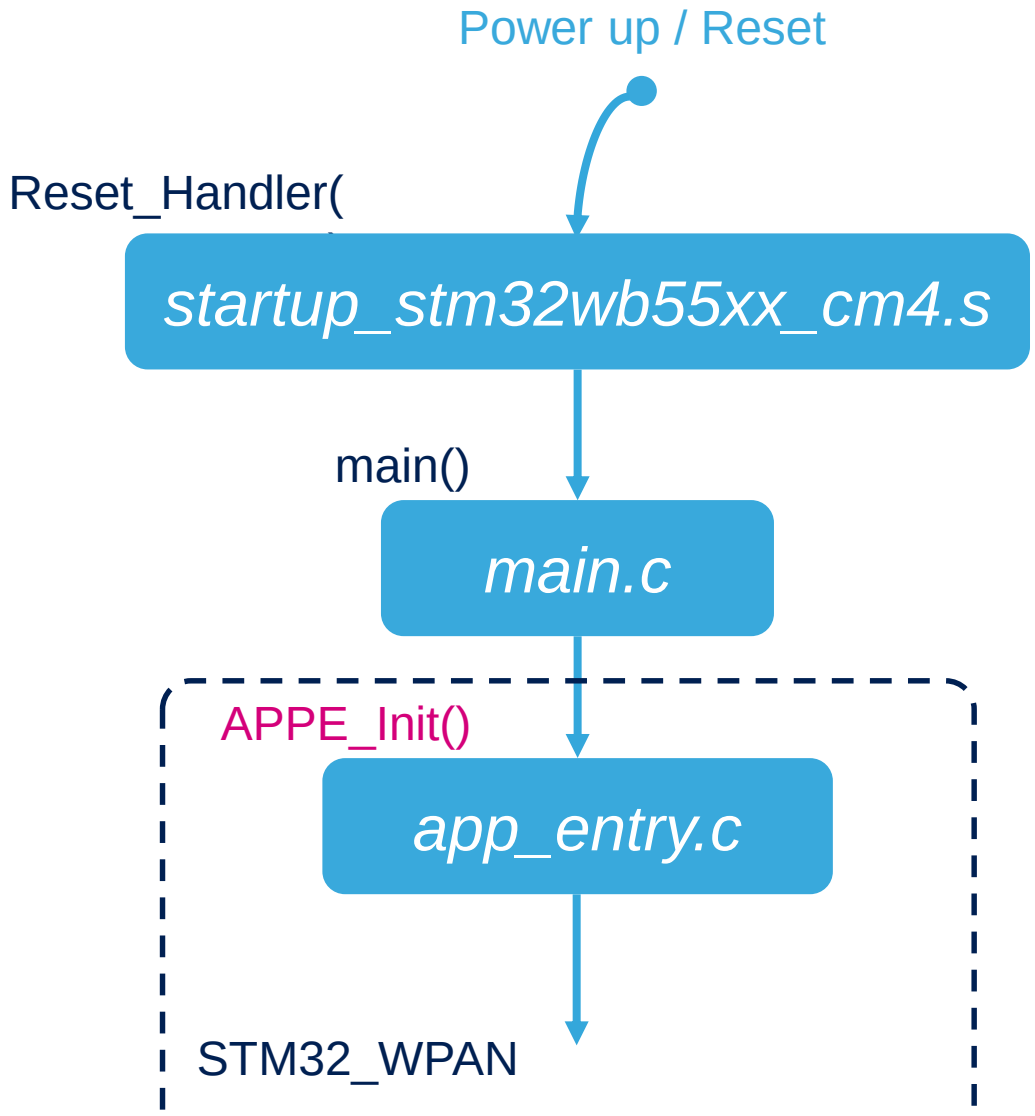
life.augmented

STM32 system blocks and IPs initialization

- Reset_Handler
  - stack pointer initialization
  - Variables initialization (SRAM memory)
  - SystemInit() call

    → main()
- main()
  - MCU HW initialization
    - RCC (clock), GPIO, RTC, I2C3,…

      → APPE_Init()



Add the HSE tuning

stm32wbxx_hal_msp.c

STEP1_Add_HSE_tuning.txt

ADD

user section **{ Includes @Line ~24 }**

1

```
/* USER CODE BEGIN Includes */
#include "otp.h"
/* USER CODE END Includes */
```

HAL_MspInit(…) user section **{ MspInit 0 @Line ~67}**

2

```
void HAL_MspInit(void)
{
/* USER CODE BEGIN MspInit 0 */
  #warning "Following code is valid only for P-NUCLEO-WB55 boards and should be re-implemented depending on the target HW and HSE capacitor tuning value storage location."
  OTP_ID0_t * p_otp;

  /**
    * Read HSE_Tuning from OTP
    */
  p_otp = (OTP_ID0_t *) OTP_Read(0);
  if (p_otp)
  {
    LL_RCC_HSE_SetCapacitorTuning(p_otp->hse_tuning);
  }
/* USER CODE END MspInit 0 */

}
```

New feature of STM32WB Target HW specific

AN5042

life.augmented

Add STM32_WPAN ISRs

stm32wbxx_it.c

ADD

1

user section **{ Includes @Line ~25 }**

```
/* USER CODE BEGIN Includes */
#include "app_common.h"
/* USER CODE END Includes */
```

2

user section **{ 1 @Line ~215 }**

```
/* USER CODE BEGIN 1 */
/**
  * @brief This function handles RTC wake-up interrupt through EXTI line 19.
  */
void RTC_WKUP_IRQHandler(void)
{
  HW_TS_RTC_Wakeup_Handler();
}
/**
  * @brief This function handles IPCC RX occupied interrupt.
  */
void IPCC_C1_RX_IRQHandler(void)
{
  HW_IPCC_Rx_Handler();
}
/**
  * @brief This function handles IPCC TX free interrupt.
  */
void IPCC_C1_TX_IRQHandler(void)
{
  HW_IPCC_Tx_Handler();
}
/* USER CODE END 1 */
```

STEP2_Add_STM32_WPAN_ISRs.txt

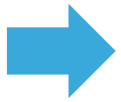
These interrupt handlers and callbacks implemented in STM32_WPAN modules are currently not generated by STM32CubeMX when STM32_WPAN is in use

IPCC managed by STM32_WPAN completely

life.augmented

Typical simple application architecture


```
main() {
…
   while(1) {
      switch(event):
      case EVENT1:
        Task1();
        clear_EVENT1();
      …
      case EVENTX:
        TaskX();
        clear_EVENTX();
      case IDLE:
        Enter_Low_Power_Mode();
      default:
        break;
   }
}
```

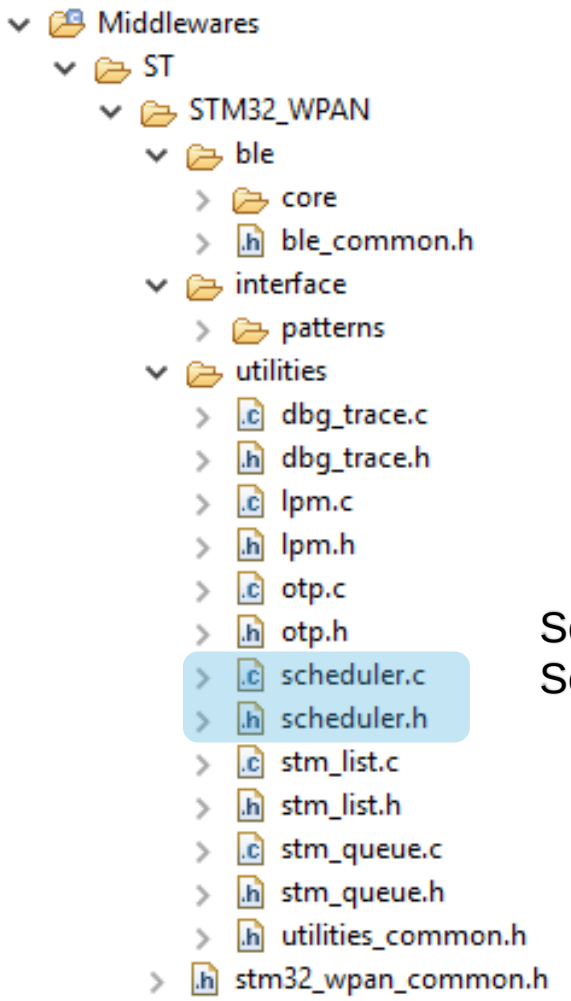


Sequencer (Simple Scheduler) as a basic task manager

Simple Scheduler

The scheduler provides the following features:

- ✓ Up to 32 tasks registered
- ✓ Request a task to be executed
- ✓ Pause and Resume a task
- ✓ Wait for a specific event (might be not blocking)
- ✓ Priority on tasks



Scheduler.c
Scheduler.h

Simple Scheduler

- Register a task to be executed in the background at any time / any place in the firmware (before it is requested to be executed)
- Enter low power mode when there is nothing to schedule
- Request the scheduler to execute a task according to priority in the background. The request may be done at any time / any place in the firmware (from interrupt handler, function, etc…)
- List of API
  - SCH_Idle()
  - SCH_Run()
  - SCH_RegTask()
  - SCH_SetTask()
  - SCH_PauseTask()
  - SCH_ResumeTask()
  - SCH_WaitEvt()
  - SCH_SetEvt()
  - SCH_IsEvtPend()
  - SCH_EvtIdle()

```
Main( void )
{
  HAL_Init();
  .  .  .
 SCH_RegTask( Id1, Task1);
  SCH_RegTask( Id2, task2);
  .  .  .
while(1)
  {
    SCH_Run(~0);
  }
}

void SCH_Idle(  void )
{
  LPM_EnterModeSelected();
}


void fct ( void )
{
```

Register tasks to be executed in the background

Enter low power mode when there is nothing to schedule

Request the scheduler to execute Task1 in the background

Request the scheduler to execute Task2 in the background

Add STM32_WPAN scheduler call

main.c

user section { **Includes @Line ~40** }

```
/* USER CODE BEGIN Includes */
#include "scheduler.h"
/* USER CODE END Includes */
```

ADD

user section { **3 @Line ~116** }

```
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  SCH_Run(~0);
}
/* USER CODE END 3 */
```

Not generated by STM32CubeMX yet when STM32_WPAN is in use.

STEP3_Add_scheduler_call.txt

OLED display functionality

Display the Device name

STEP4_Add_OLED_init.txt

app_ble.c

ADD

1

user section **{ Includes @Line ~40 }**

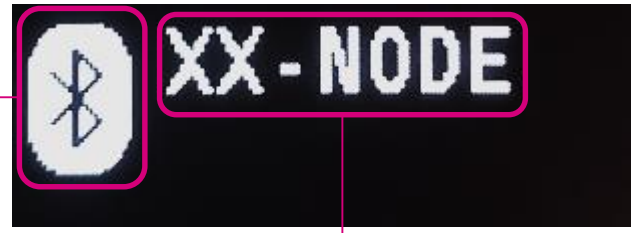
```
/* USER CODE BEGIN Includes */
#include "hal_lcd.h"
/* USER CODE END Includes */
```

2

user section **{ APP_BLE_Init_1 @Line ~381 }**

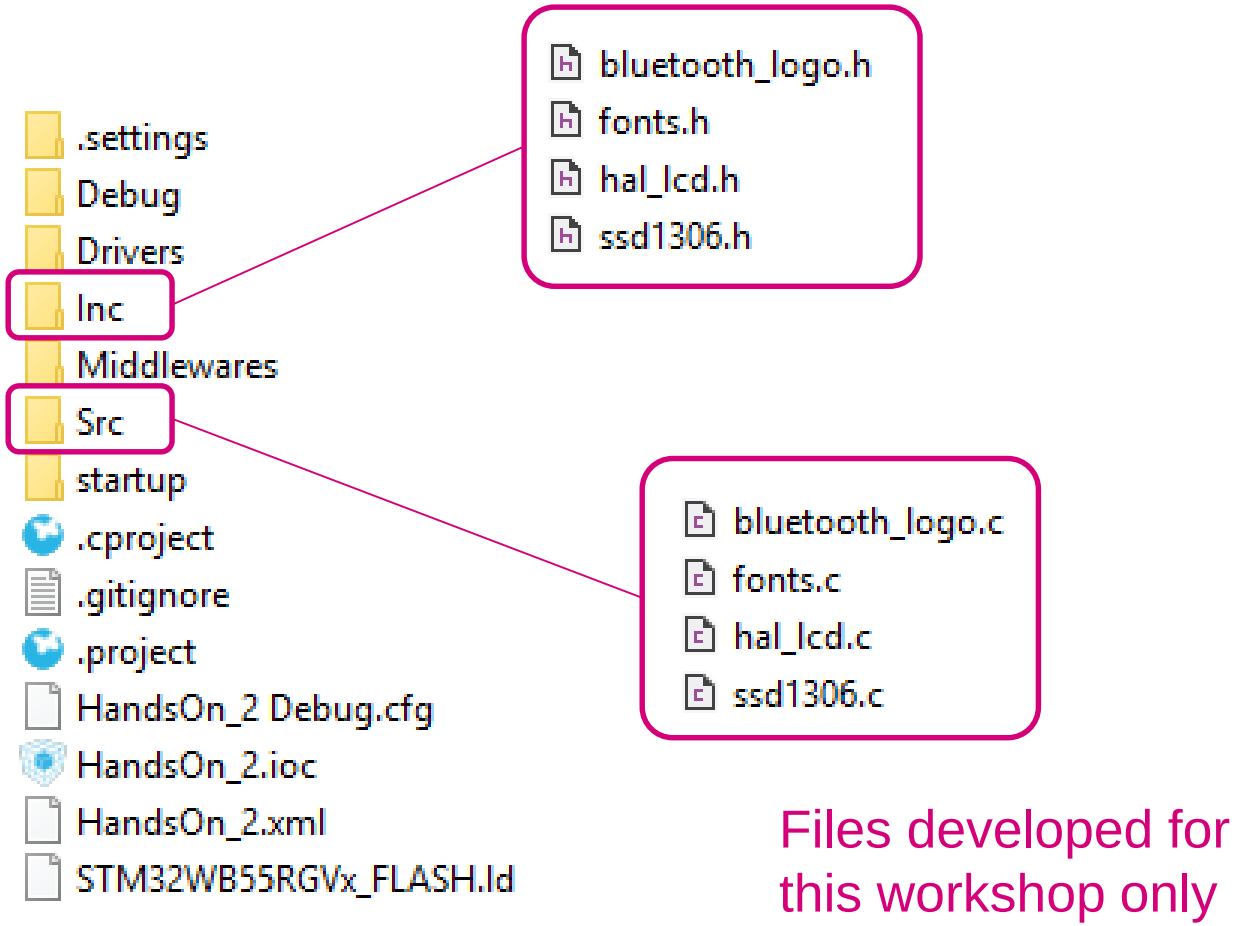
```
/* USER CODE BEGIN APP_BLE_Init_1 */
  /* Initialize the LCD */
  LCD_Init();
  /* Display the application icon */
  LCD_BLE_PrintLogo();
  /* Display the local device name */
  LCD_BLE_PrintLocalName(local_name);
/* USER CODE END APP_BLE_Init_1 */
```

According to the STM32_WPAN architecture, LCD_Init() shall be called from app_entry.c. We will put in app_ble.c just for simplicity.



life.augmented

OLED display driver files



Files developed for this workshop only

Add the green LED blinking

app_ble.c : Line ~631

STEP5_Add_GREEN_LED_blinking.txt

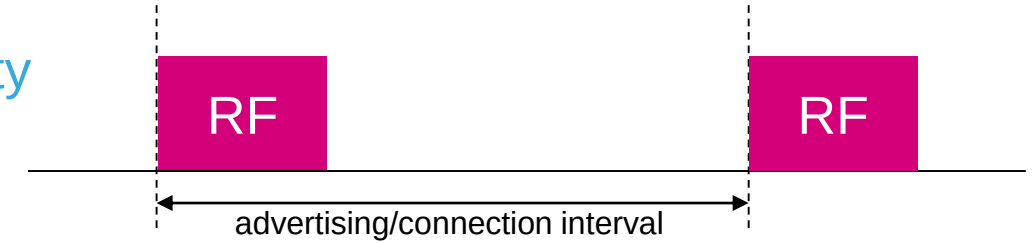
ADD

SVCCTL_App_Notification(…) user section **{ RADIO_ACTIVITY_EVENT }**

```
/* USER CODE BEGIN RADIO_ACTIVITY_EVENT */
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, GPIO_PIN_SET);
HAL_Delay(5);
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, GPIO_PIN_RESET);
/* USER CODE END RADIO_ACTIVITY_EVENT */
```

Generate 5ms flash with BLUE LED.

RADIO_ACTIVITY_EVENT
Triggered after every Radio RF activity finishes
Event mask configurable
(ADVERTISE, SCAN, CONNECTION)



RADIO_ACTIVITY_EVENT enabled in app_conf.h

life.augmented

Test the functionality

1 Build

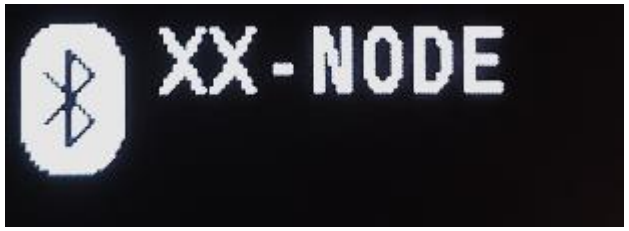
or Ctrl+B

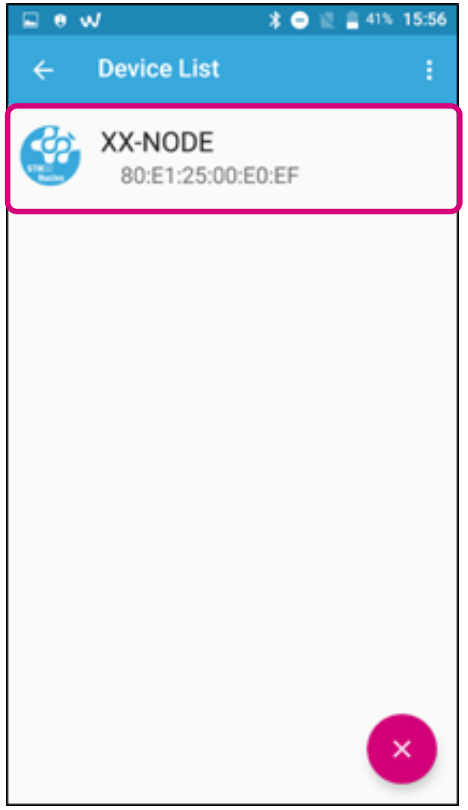
2 Debug

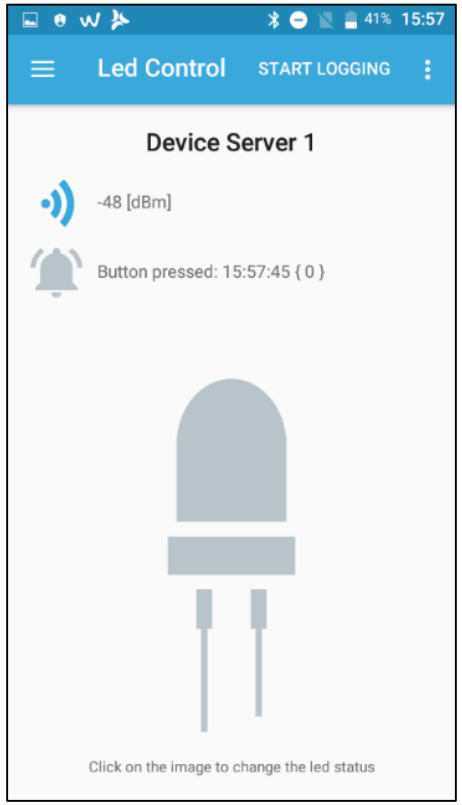
F11

3 Resume

F8







GREEN LED blinking period changes when advertising vs. connected

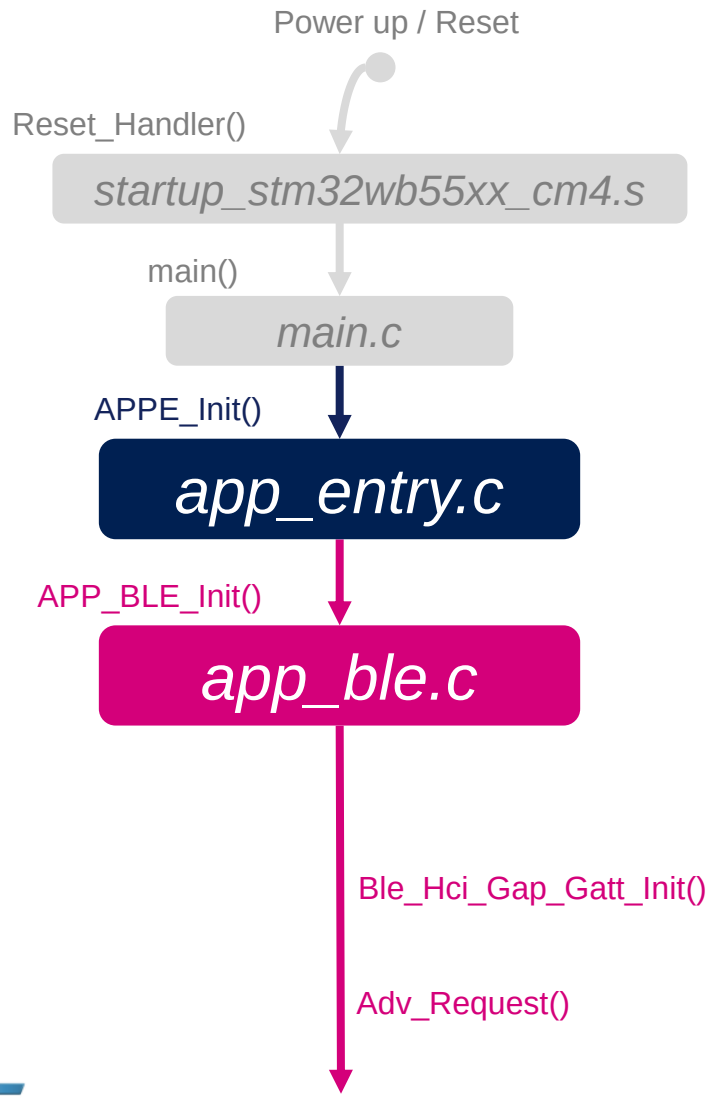
Advertising stops after 60sec if link not established (GREEN LED stops blinking)

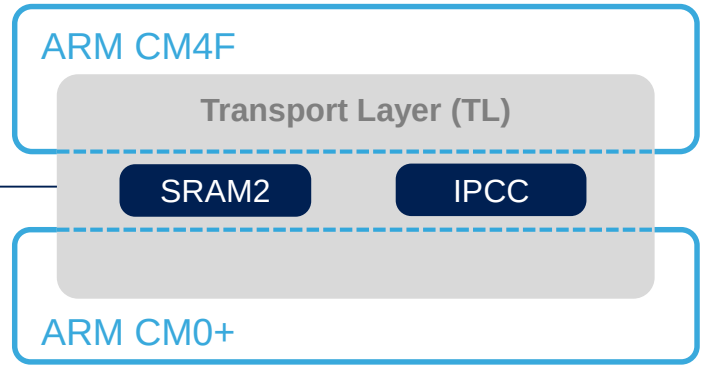
Reset for restart

DISCONNECT

life.augmented

BLE Application Initialization





Initialize **TL** (boot up CM0+) + other app specific HW

→ Wait for initialization done **given boot up sequence**

Initialize the BLE stack

(**GATT** and **GAP** initialization), advertising mode control

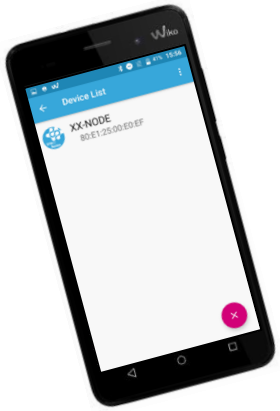
→ Set TX Output Power
→ GATT init
→ GAP init
→ **Set discoverable (start advertising)**

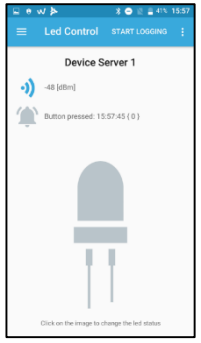
Generated code

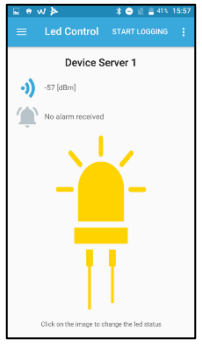
life.augmented

What is running now?

Central **GATT Client**







P2P Client **P2P Server**

Scanning Mode

Advertising (???)

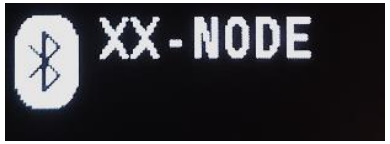
Advertising (???)

Connect Request

Link (connection) established

Advertising Mode

Peripheral **GATT Server**



How do we filter out on the central side just the devices running our simple example application?

life.augmented

BLE Advertising data

Over-The-Air BLE Packet

Length	1 byte	4 bytes	2~257 bytes	3 bytes
Name	Preamble	Access Address	Protocol Data Unit (PDU)	CRC
Value	10101010b	0xXXXXXXXX	0xXX……XX	0xXXXXXX
	Don’t care now			Don’t care now

Advertising PDU

Length	2 bytes	6 bytes	0~31 bytes
Name	Header	Advertising Address	Advertising Data
Value	0xXXXX	0xXXXXXXXXXXX	0xXX……XX
	Don’t care now		**Interesting for us**

Data PDU

Not under scope now

BLE Advertising data

Advertising PDU

Length	2 bytes	6 bytes	0~31 bytes
Name	Header	Advertising Address	Advertising Data
Value	0xXXXX	0xXXXXXXXXXXXX	0xXX……XX

Don’t care now

AD structure	AD structure	AD structure	·····	AD structure

AD structure format

Length	1 byte	1 byte	(Length – 1) bytes
Name	Length	Type	Data

Several types defined, e.g.:
- 0x09 – Complete local name
- 0xFF – Manufacturer specific data

Advertised complete local name

AD structure of complete local name

Length	1 byte	1 byte	(Length – 1) bytes
Name	Length	Type	Complete local name
Value	0xXX	0x09	0xXXXX……XX

e.g. { 'X', 'X', '-', 'N', 'O', 'D', 'E' }

app_ble.c

Private variables

```
static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME,'X','X','-','N','O','D','E' };
```

life.augmented

Advertising start

app_ble.c

Adv_Request(…) { } called at the end of **APP_BLE_Init**(…) { }

```
static void Adv_Request( void )
{
    ret = aci_gap_set_discoverable(
            ADV_IND,
            Min_Interval,
            Max_Interval,
            PUBLIC_ADDR,
            NO_WHITE_LIST_USE, /* use white list */
            sizeof(local_name),
            (uint8_t*) local_name,
            BleApplicationContext.BleApplicationContext_legacy.advtServUUIDlen,
            BleApplicationContext.BleApplicationContext_legacy.advtServUUID,
            0,
            0);
}
```

Start advertising ⟶ *aci_gap_set_discoverable(…);*

BlueST Protocol

AD structure of our BlueST Protocol

Length	1 byte	1 byte	1 byte	1 byte	4 bytes	6 bytes
Name	Length	Type	Protocol Version	Device Id	Feature Mask	Device MAC (optional)
Value	0x07/0xD	0xFF	0x01	0xXX	0xXXXXXXXX	0xXXXXXXXXXXXX

16-bit Company ID provided by Bluetooth SIG should be used here normally

0x00 for a generic device
0x01 - STEVAL-WESU1 board
0x02 - STEVAL-STLKT01V1 (SensorTile) board
0x03 - STEVAL-BCNKT01V1 (BlueCoin) board
0x04 - STEVAL-IDB008V1/2 (BlueNRG-2) board
0x05 - STEVAL-BCN002V1B (BlueNRG-Tile) board
0x80 to 0x8A for various functional packs for Nucleo boards

BlueST Protocol description (@github)

life.augmented

app_ble.c

Manufacturer specific data

user code section **{ PV }**

```
/* USER CODE BEGIN PV */

/* Manufacturer specific data */
uint8_t manuf_data[14] = { sizeof(manuf_data)-1, /* AD_RECORD Length */
                           AD_TYPE_MANUFACTURER_SPECIFIC_DATA, /* AD_RECORD Type */
                           0x01, /* Protocol Version */
                           CFG_DEV_ID_P2P_SERVER1, /* Device Id */
                           0x00, /* GROUP A Feature  */
                           0x00, /* GROUP A Feature */
                           0x00, /* GROUP B Feature */
                           0x00, /* GROUP B Feature */
                           0x00, /* BLE MAC start -MSB */
                           0x00,
                           0x00,
                           0x00,
                           0x00,
                           0x00  /* BLE MAC stop */
                           };

/* USER CODE END PV */
```

BlueST Protocol version to 0x01
1st byte of manufacturer specific data
(to identify BlueST protocol)

The Device ID is 0x83
2nd byte of manufacturer specific data
P2P Server 1
(according to BlueST protocol)

life.augmented

Advertising data update

app_ble.c

Adv_Request(…) { } called at the end of **APP_BLE_Init**(…) { }

```
static void Adv_Request( void )
{
  ┆
  /* Update Advertising data */
  ret = aci_gap_update_adv_data(sizeof(manuf_data), (uint8_t*) manuf_data);
  ┆
}
```

Start advertising ⟶ *aci_gap_set_discoverable(…);*

Update advertising data ⟶ *aci_gap_update_adv_data(…);*

Advertising stop

app_ble.c

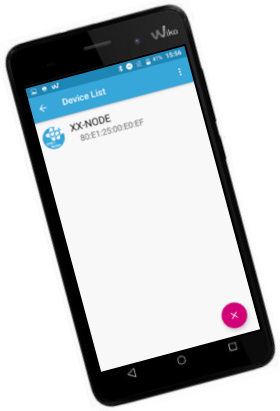
Adv_Cancel(…) { } called after defined timeout (60 sec – fully up to the user) { }

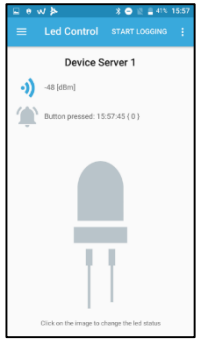
```
static void Adv_Cancel( void )
{

  result = aci_gap_set_non_discoverable();

}
```

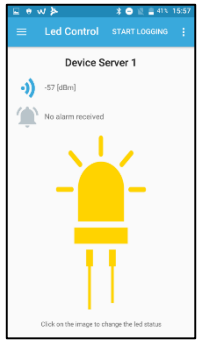
Start advertising ⟶ *aci_gap_set_discoverable(…);*
Update advertising data ⟶ *aci_gap_update_adv_data(…);*
Stop advertising ⟶ *aci_gap_set_non_discoverable(…);*

What is running now?

Central **GATT Client**



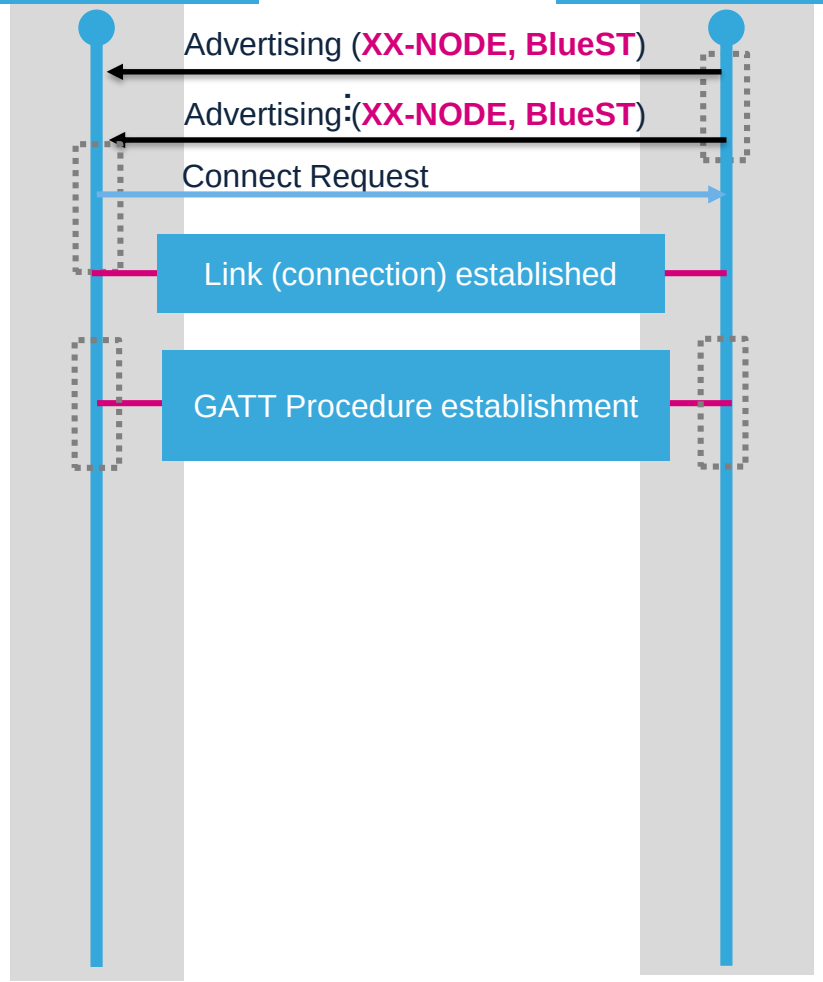




Scanning Mode

ATT Handle Discovery

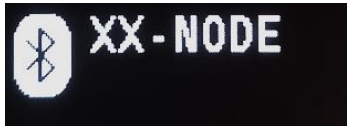
P2P Client **P2P Server**



Peripheral **GATT Server**

Advertising Mode

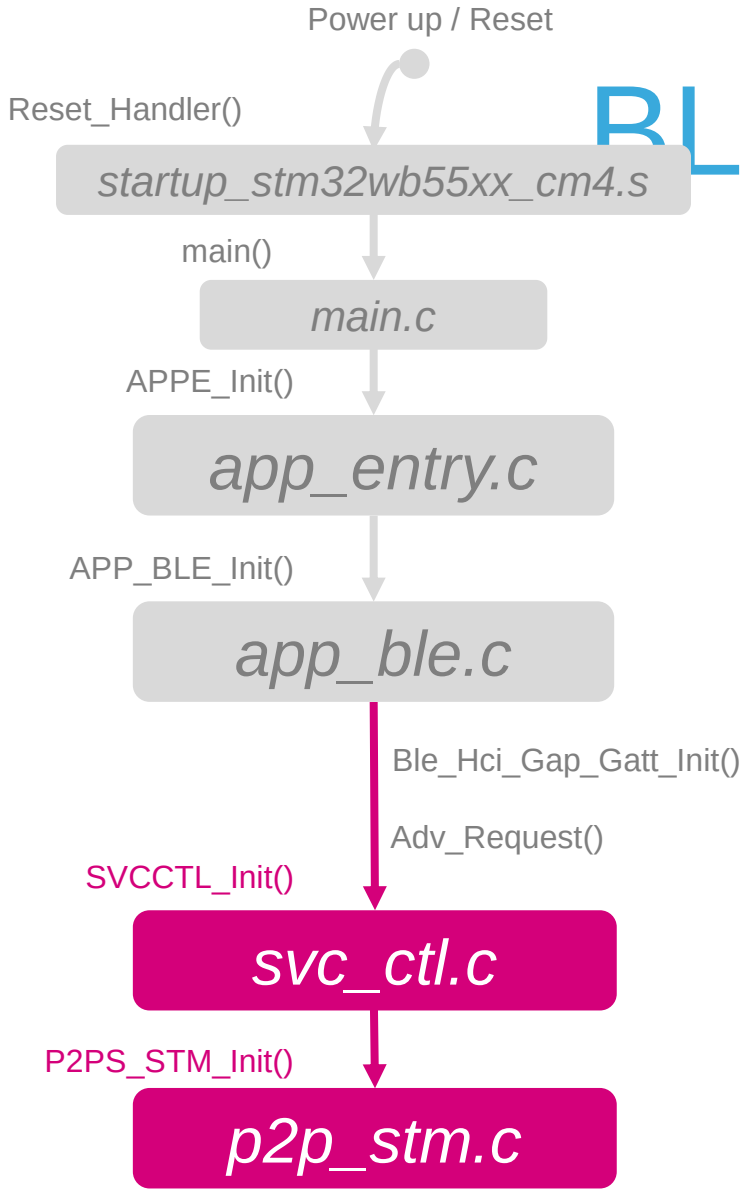
ATT Services & Characteristics

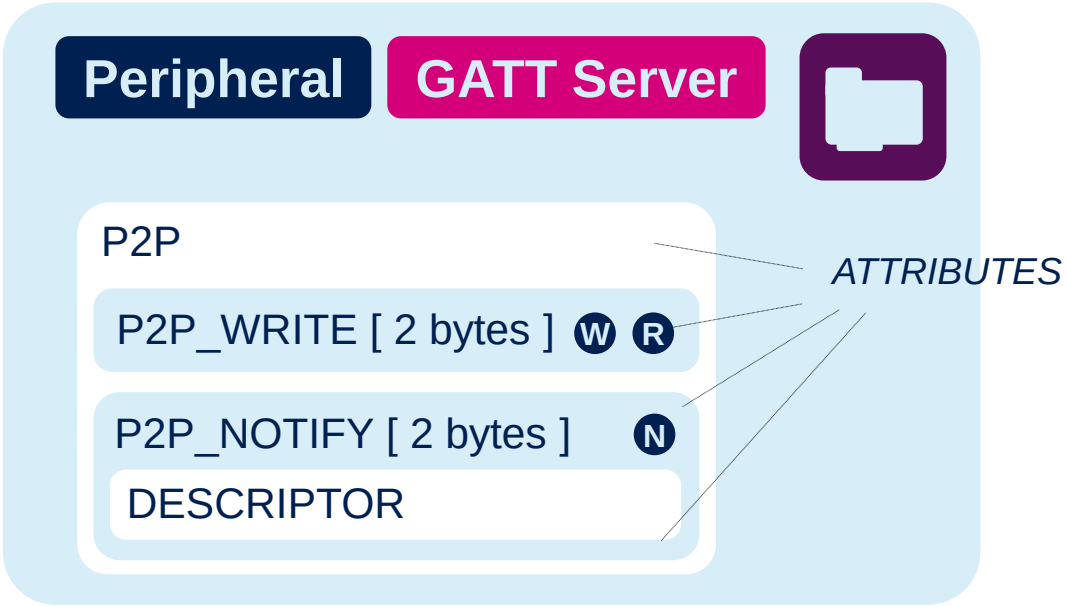


GATT Server initialized already too, where and how?

life.augmented

BLE Application Initialization





Add **GATT** services and characteristics

Generated code

Add P2P STM service and characteristics

Add Service → *aci_gatt_add_service(…)*

Add Characteristic → *aci_gatt_add_char(…)*

life.augmented

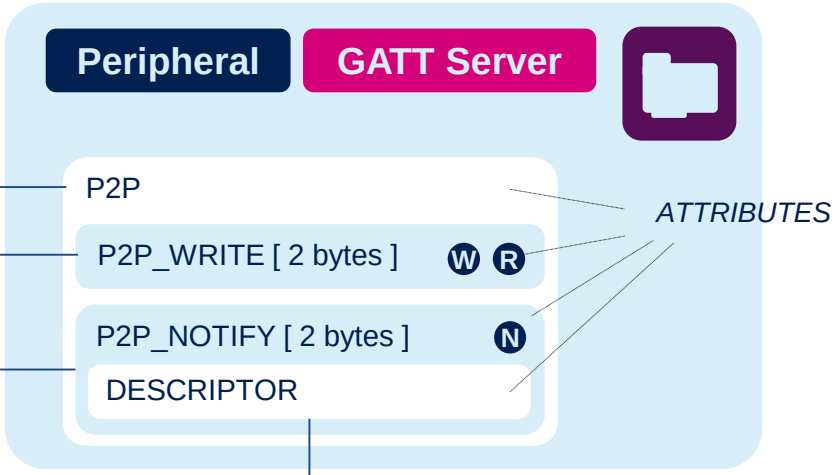
P2P_STM Service overview

UUID (hex)	0000FE40-CC7A-482A-984A-7F2ED5B3E58F (proprietary)
Type	PRIMARY SERVICE

UUID	0000FE41-8E22-4541-9D4C-21EDAE82ED19 (proprietary)		
Properties	WRITE NO RESPONSE | READ		
Value	Byte	1 (LED state)	0 (Device number)
		0x00 – LED on 0x01 – LED off	0x00 - all 0x01~0x06 – P2P Server 1~6

UUID (hex)	0000FE42-8E22-4541-9D4C-21EDAE82ED19 (proprietary)	
Properties	NOTIFY	
Value	Byte	1 (Button status)	0 (Device number)
		0x00 – button pressed 0x01 – button released	0x01~0x06 – P2P Server 1~6

UUID (hex)	2902 (Client Characteristic Configuration)		
Value	Bit	15~2	1 (Indication state)	0 (Notification state)
		Reserved for future use	0 – Indications disabled 1 – Indications enabled	0 – Notifications disabled 1 – Notifications enabled



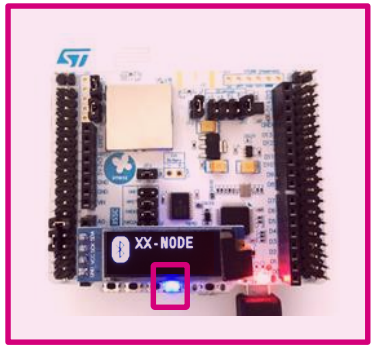
life.augmented

What to do **next**?

Central **GATT Client**

P2P Client **P2P Server**

Peripheral **GATT Server**



Scanning Mode

Advertising Mode

Advertising (XX-NODE, BlueST)

Advertising (XX-NODE, BlueST)

Connect Request

Link (connection) established

ATT Handle Discovery

GATT Procedure establishment

ATT Services & Characteristics

Enable Notification

Central-to-Peripheral communication **(Control SLAVE)**

Write (0x00) – Led ON

Write (0x01) – Led OFF

Tap Button in the phone app to write new characteristic #2 value

Characteristic already being written, but the event and value not processed

Control the blue LED upon write characteristic event

life.augmented

P2P Server app GATT event handler

p2p_server_app.c

```
void P2PS_STM_App_Notification(P2PS_STM_App_Notification_evt_t *pNotification)
{
  switch(pNotification->P2P_Evt_Opcode)
  {
    case P2PS_STM__NOTIFY_ENABLED_EVT:
      ┆
      break;
    case P2PS_STM_NOTIFY_DISABLED_EVT:
      ┆
      break;
    case P2PS_STM_WRITE_EVT:
      /* Characteristic updated, parse the payload */
      ┆
      break;
    default:
    break;
  }
  return;
}
```

EVT_BLUE_GATT_ATTRIBUTE_MODIFIED GATT events propagated from ***PeerToPeer_Event_Handler(…)*** registered @SVCCTL during ***P2PS_STM_Init()***

P2P_WRITE characteristic value changed

Atrribute modified by client ⟶ *EVT_BLUE_GATT_ATTRIBUTE_MODIFIED*

Add the blue LED control

STEP6_Add_BLUE_LED_control.txt

if 2nd byte of P2P_WRITE characteristic value is 0x01 → Turn the blue LED ON

p2p_server_app.c : Line ~85

P2PS_STM_App_Notification(…) user section **{ P2PS_STM_WRITE_EVT }**

```
void P2PS_STM_App_Notification(P2PS_STM_App_Notification_evt_t *pNotification)
{
    ⋮
/* USER CODE BEGIN P2PS_STM_WRITE_EVT */
  if(pNotification->DataTransfered.pPayload[1] == 0x01) {
    HAL_GPIO_WritePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin, GPIO_PIN_SET);
  }
  else {
    HAL_GPIO_WritePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin, GPIO_PIN_RESET);
  }
/* USER CODE END P2PS_STM_WRITE_EVT */
    ⋮
}
```

ADD

Turn the blue LED OFF otherwise

1st byte of the value as “don’t care” now

life.augmented

Test the functionality

1 Build

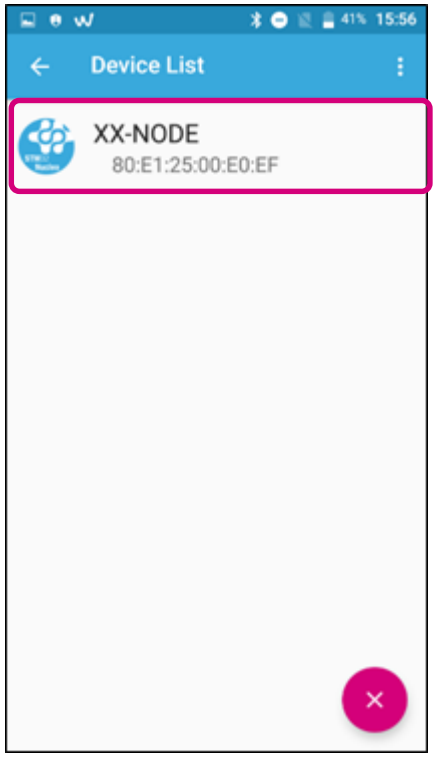
or Ctrl+B

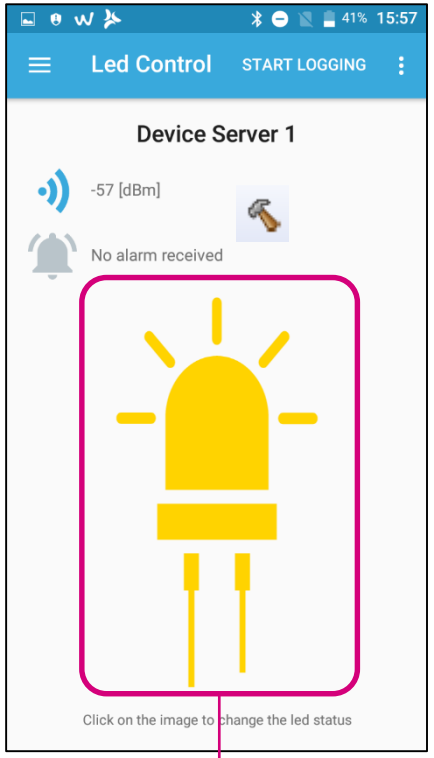
2 Debug

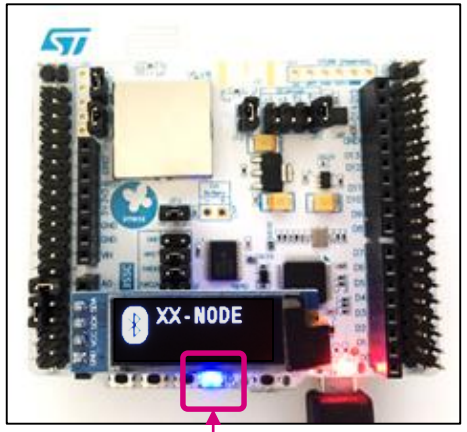
F11

3 Resume

F8

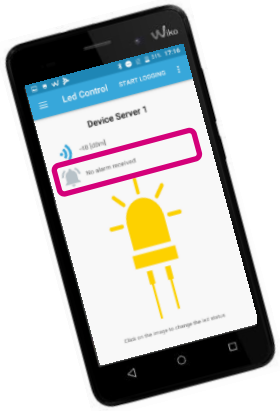






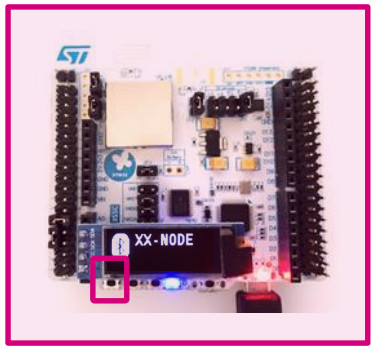
What to do **next**?

Central **GATT Client**



P2P Client **P2P Server**

Peripheral **GATT Server**



Scanning Mode

Advertising Mode

Advertising (XX-NODE, BlueST)

Advertising (XX-NODE, BlueST)

Connect Request

Link (connection) established

ATT Handle Discovery

GATT Procedure establishment

ATT Services & Characteristics

Enable Notification

Central-to-Peripheral communication **(Control SLAVE)**

Write (0x01) – Led ON

Write (0x00) – Led OFF

Update characteristic value upon SW1 button press

Tap Button in the phone app to write new characteristic #2 value

Push SW1 to update characteristic #1 value (notify value change)

Notification (0x00) – 1st Button press

Notification (0x00) – 2nd Button press

Peripheral-to-Central communication **(Notify MASTER)**

life.augmented

Implement Send Notification function

p2p_server_app.c : Line ~55 user section { PFP }

1

```
/* USER CODE BEGIN PFP */
static void P2PS_Send_Notification(void);
/* USER CODE END PFP */
```

ADD

p2p_server_app.c : Line ~150 user section { FD }

2

```
/* USER CODE BEGIN FD */
static void P2PS_Send_Notification(void)
{
  /* Update P2P_NOTIFY characteristic */
  P2PS_STM_App_Update_Char(P2P_NOTIFY_CHAR_UUID, 0x00);

  return;
}
/* USER CODE END FD */
```

STEP7_Add_P2PS_Send_Notification.txt

p2p_stm.c

Update Characteristic value ⟶ *aci_gatt_update_char_value(…)*

life.augmented

Register Send Notification function as a task

STEP8_Add_P2PS_notify_task.txt

p2p_server_app.c : Line ~141

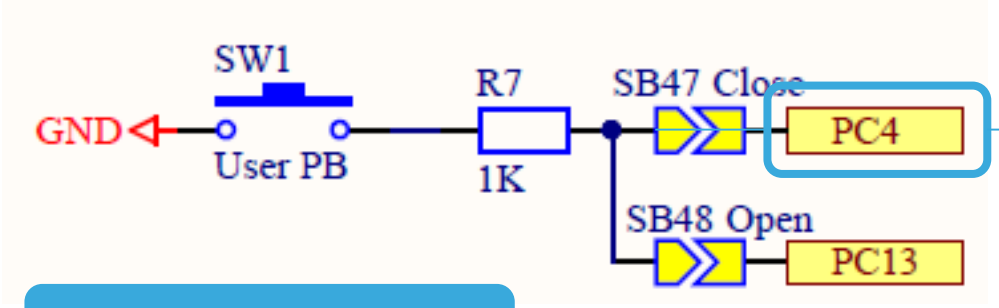
P2PS_APP_Init(…) user section **{ P2PS_APP_Init }**

```
void P2PS_APP_Init(void)
{
/* USER CODE BEGIN P2PS_APP_Init */
  SCH_RegTask( CFG_TASK_SW1_BUTTON_PUSHED_ID, P2PS_Send_Notification );

/* USER CODE END P2PS_APP_Init */
}
```

ADD

From where we should trigger this task now?

EXTI on GPIO pin connected to button SW1



PC4 pin already configured as following:

- GPIO_EXTIx
- BUTTON_SW1 label
- Internal Pull-Up enabled
- Falling-Edge detection activated
- EXTI Line 4 interrupt enabled in NVIC

stm32wbxx_it.c

```
void EXTI4_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI4_IRQn 0 */

  /* USER CODE END EXTI4_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_4);
  /* USER CODE BEGIN EXTI4_IRQn 1 */

  /* USER CODE END EXTI4_IRQn 1 */
}
```

stm32wbxx_hal_gpio.c

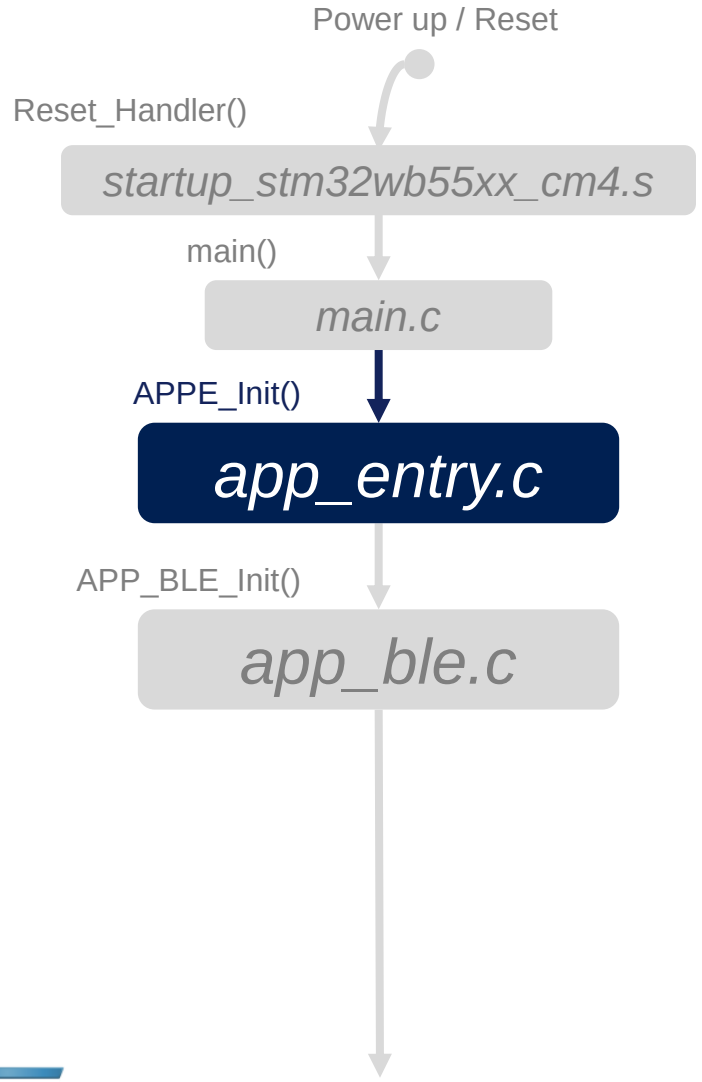
```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != 0x00u)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
```

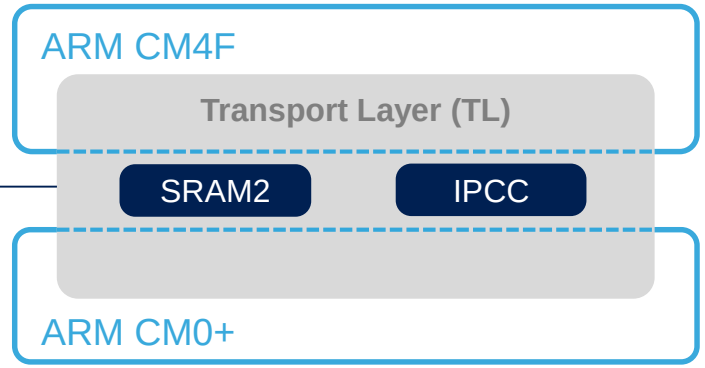
GENERATED CODE

TO BE IMPLEMENTED IN END USER CODE

life.augmented

Implement GPIO EXTI callback, where???





Initialize **TL** (boot up CM0+) + other app specific HW

→ Wait for initialization done **given boot up sequence**

We need to place somewhere HAL_GPIO_EXTI_IRQHandler(…) callback.

app_entry.c file is designed by the middleware architecture to hold the additional HW related functions. Also, in case you would like to use other EXTI channels for other purposes, the HAL callback function can be only one in the complete application code.

Call Send Notification from SW1 Button EXTI callback

app_entry.c : Line ~107

STEP9_Add_EXTI_callback.txt

ADD

user code section **{ FD }**

```
/* USER CODE BEGIN FD */

void HAL_GPIO_EXTI_Callback( uint16_t GPIO_Pin )
{
  switch (GPIO_Pin)
  {
  case BUTTON_SW1_Pin:
    SCH_SetTask(1<<CFG_TASK_SW1_BUTTON_PUSHED_ID, CFG_SCH_PRIO_0);
    break;
  default:
    break;
  }
  return;
}

/* USER CODE END FD */
```

Callback called from HAL_GPIO_EXTI_IRQHandler(…) implemented in stm32wb_hal_gpio.c and called from EXTI4_IRQHandler() in stm32wbxx_it.c

We are still in ISR !

Proper Non-Blocking implementation

life.augmented

Test the functionality

1 Build

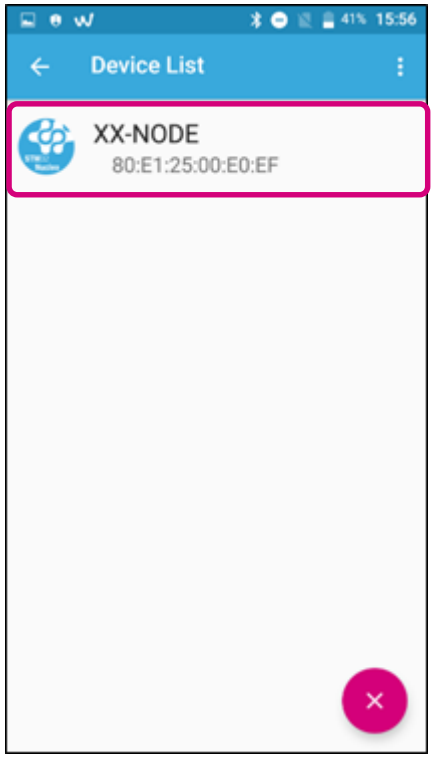
or Ctrl+B

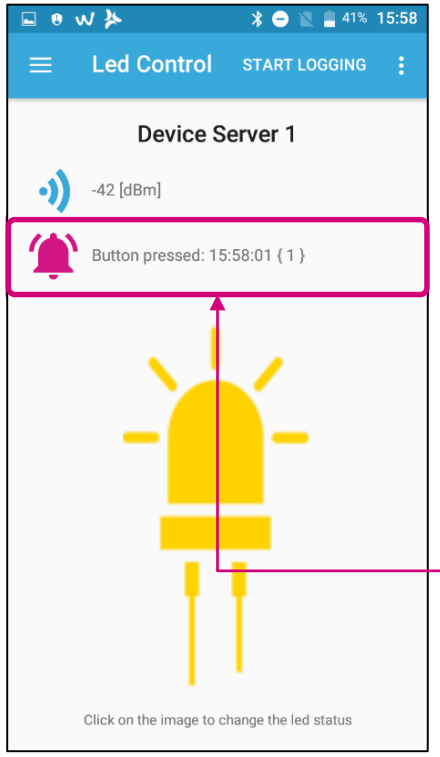
2 Debug

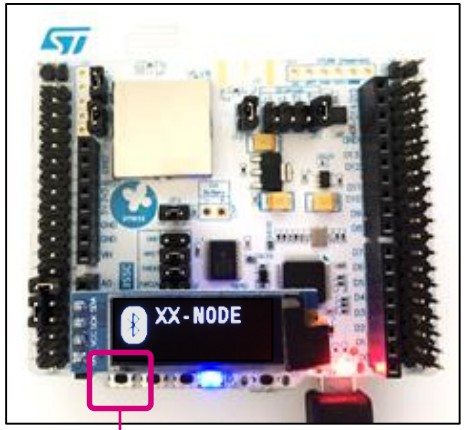
F11

3 Resume

F8







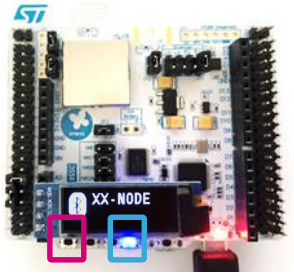
life.augmented

Finished !

Central **GATT Client**

P2P Client **P2P Server**

Peripheral **GATT Server**



Scanning Mode

Advertising Mode

Advertising (XX-NODE, BlueST)

Advertising (XX-NODE, BlueST)

Connect Request

Link (connection) established

ATT Handle Discovery

GATT Procedure establishment

ATT Services & Characteristics

Enable Notification

Central-to-Peripheral communication **(Control SLAVE)**

Write (0x01) – Led ON

Write (0x00) – Led OFF

Tap Button in the phone app to write new characteristic #2 value

Push SW1 to update characteristic #1 value (notify value change)

Notification (0x00) – 1st Button press

Notification (0x00) – 2nd Button press

Peripheral-to-Central communication **(Notify MASTER)**

life.augmented

GAP and GATT commands and events used so far

GAP

Start advertising → `aci_gap_set_discoverable(…);`
Update advertising data → `aci_gap_update_adv_data(…);`
Stop advertising → `aci_gap_set_non_discoverable(…);`

Link lost (Disconnected) → `EVT_DISCONN_COMPLETE`
Link established (Connected) → `EVT_LE_META_EVENT (EVT_LE_CONN_COMPLETE)`

GATT

Add Service → `aci_gatt_add_service(…);`
Add Characteristic → `aci_gatt_add_char(…);`
Update Characteristic value → `aci_gatt_update_char_value(…);`

Atrribute modified by client → `EVT_BLUE_GATT_ATTRIBUTE_MODIFIED`

신제품 NEW

제품 개요

Digi-Key 부품 번호	497-18384-ND
주문 가능 수량	0
제조업체	STMicroelectronics
제조업체 부품 번호	P-NUCLEO-WB55
제품 요약	BLE NUCLEO PACK INCLUDING USB DO

가격 및 조달

수량 1

497-18384-ND ▾

고객참조번호(선택 사항)

장바구니에 추가

모든 가격은 KRW 기준입니다.

수량	단가	금액
1	54,054.00000	₩54,054

표시된 주문 수량보다 더 많은 수량이 필요한 경우 견적 요청서를 제출해 주세요.

문서 및 미디어

규격서	P-NUCLEO-WB55 Data Brief
비디오 파일	Why pick the STM32WB for your next design?
주요제품	STM32WB55 P-NUCLEO-WB55 Platform

제품 특성

모두 선택 ☐

종류	RF/IF 및 RFID RF 평가 및 개발 키트, 기판	○ ◉
제조업체	STMicroelectronics	☐
계열	-	☐

life.augmented

Releasing Your Creativity

/STM32 @ST_World ST community

life.augmented

www.st.com/stm32wb