

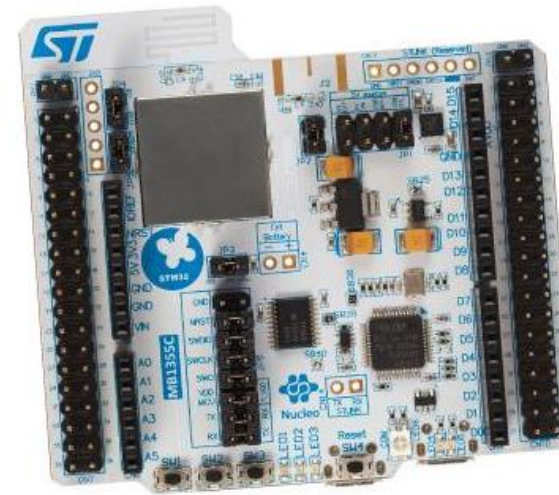
STM32WB MCU series

Excellence in **Connectivity**



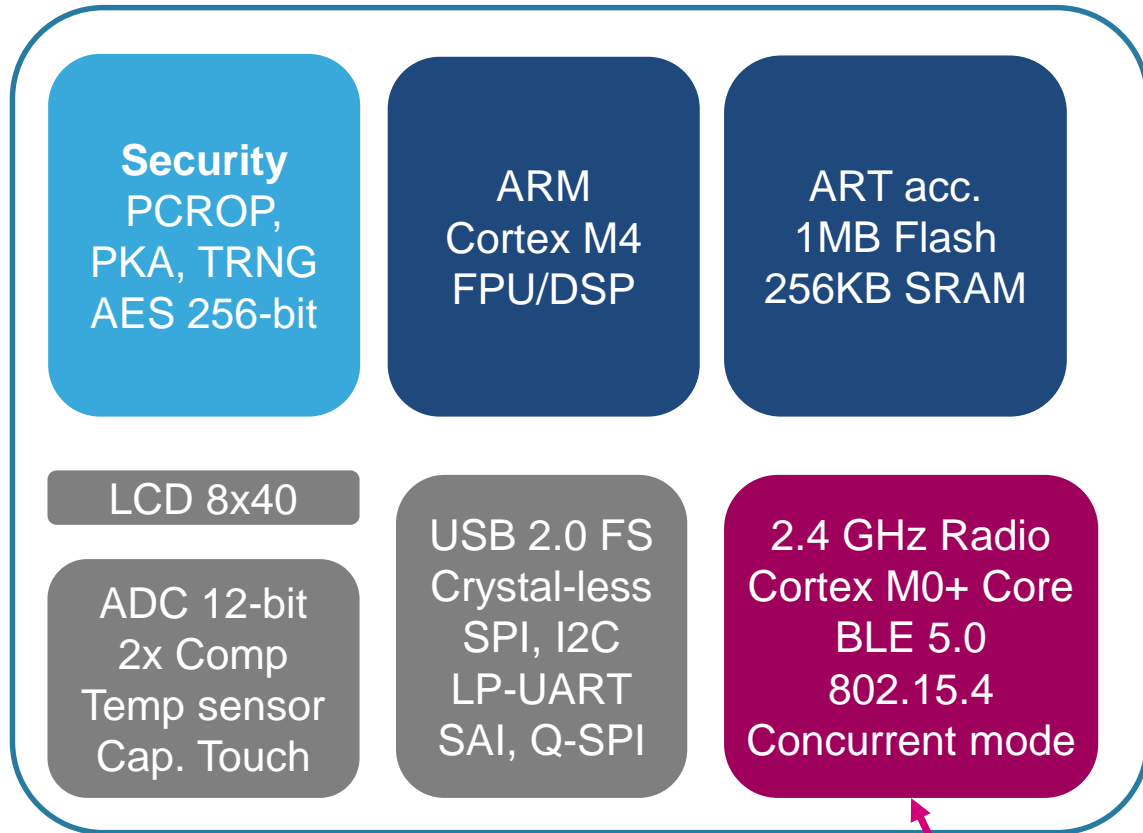


About the STM32WB





About the STM32WB



KEY FEATURES

- **2 independent cores for real time execution**
- **Ultra-low-power consumption**
 - 50 μ A/MHz Active mode (at 3.0V)
 - 1.8 μ A Stop mode (Radio in standby + 256KB RAM)
 - < 50 nA Shutdown mode
- **Peripherals**
 - 2xI²C, 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**

**Independent
Sub-system RF
2.4GHz**



All in one MCU - Full flexibility control



- 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



Lighting



Fleet maintenance

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

- 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**



Industrial devices



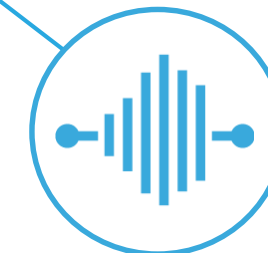
Fitness/Healthcare

- **Multipoint** BLE 5 connections
- Small form factor design with **CSP100 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

- **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



Beaconing



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





Make the Choice of STM32WB Series

The 7 keys points to make the difference



**Open 2.4 GHz radio
Multi-protocol**



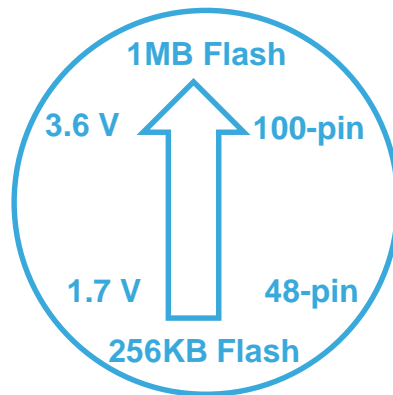
**Dual-core / Full control
Ultra-low-power**



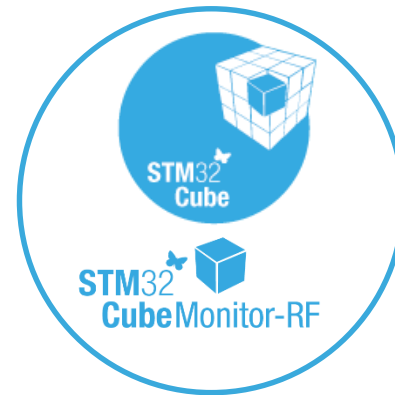
IoT Protection ready



**Massive integration
Cost saving**



A large offer



**Advanced RF tool, Energy control
with C code generation**

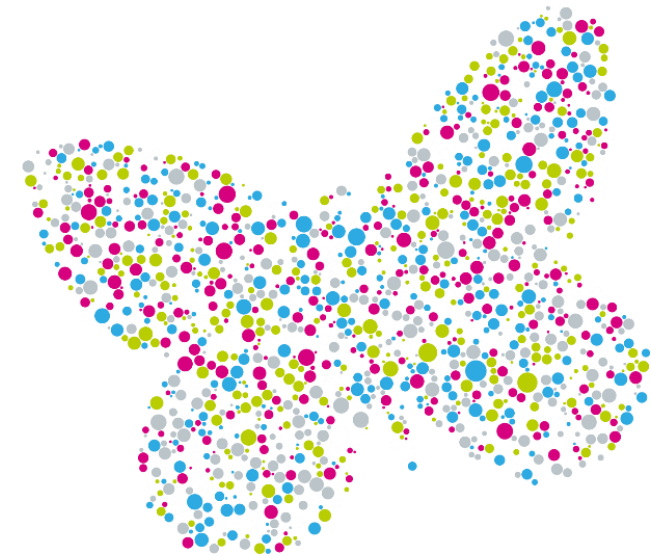


No matter what!



Great investment

12 product series / More than 50 product lines



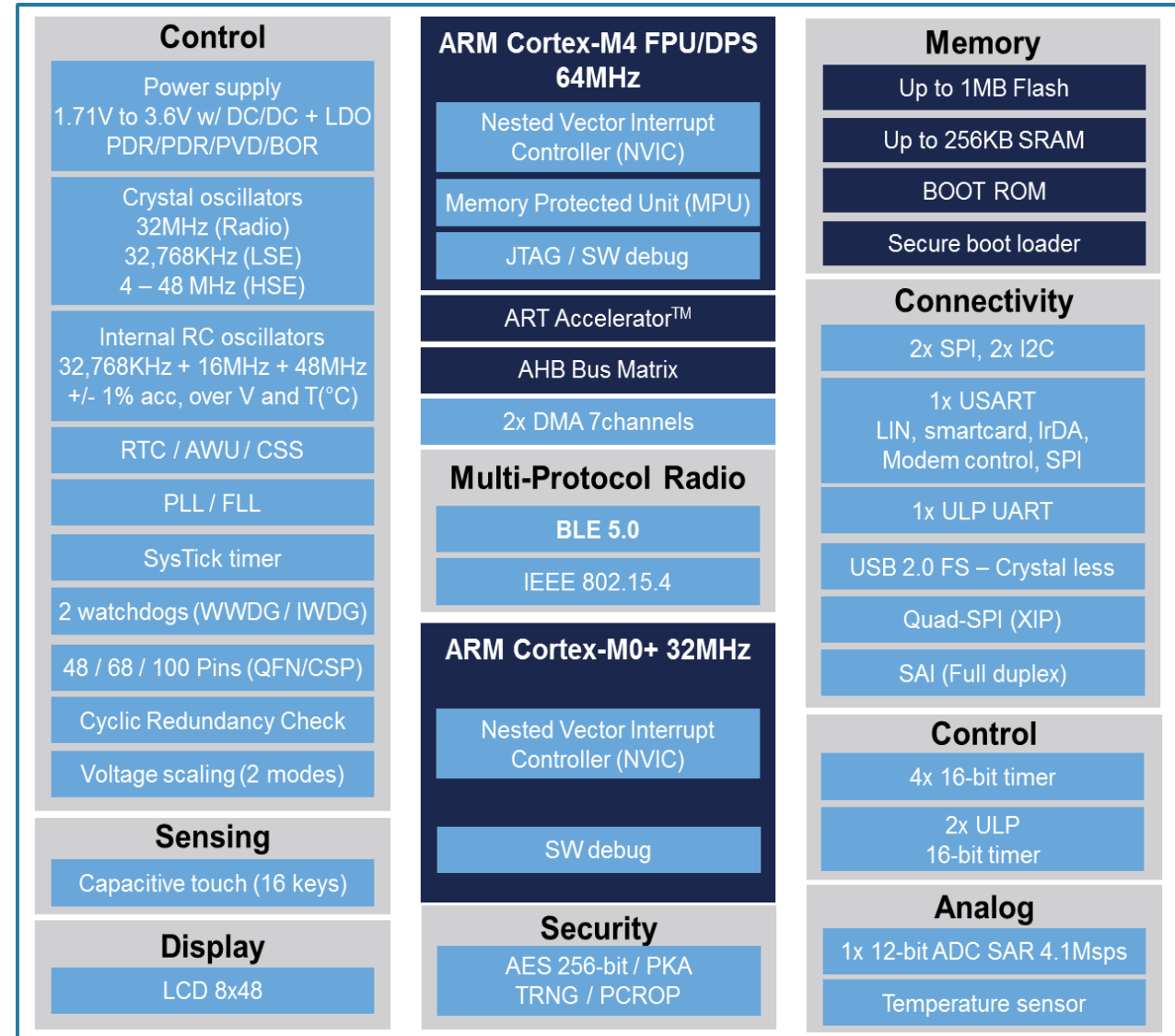
More than
40,000 customers





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





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)

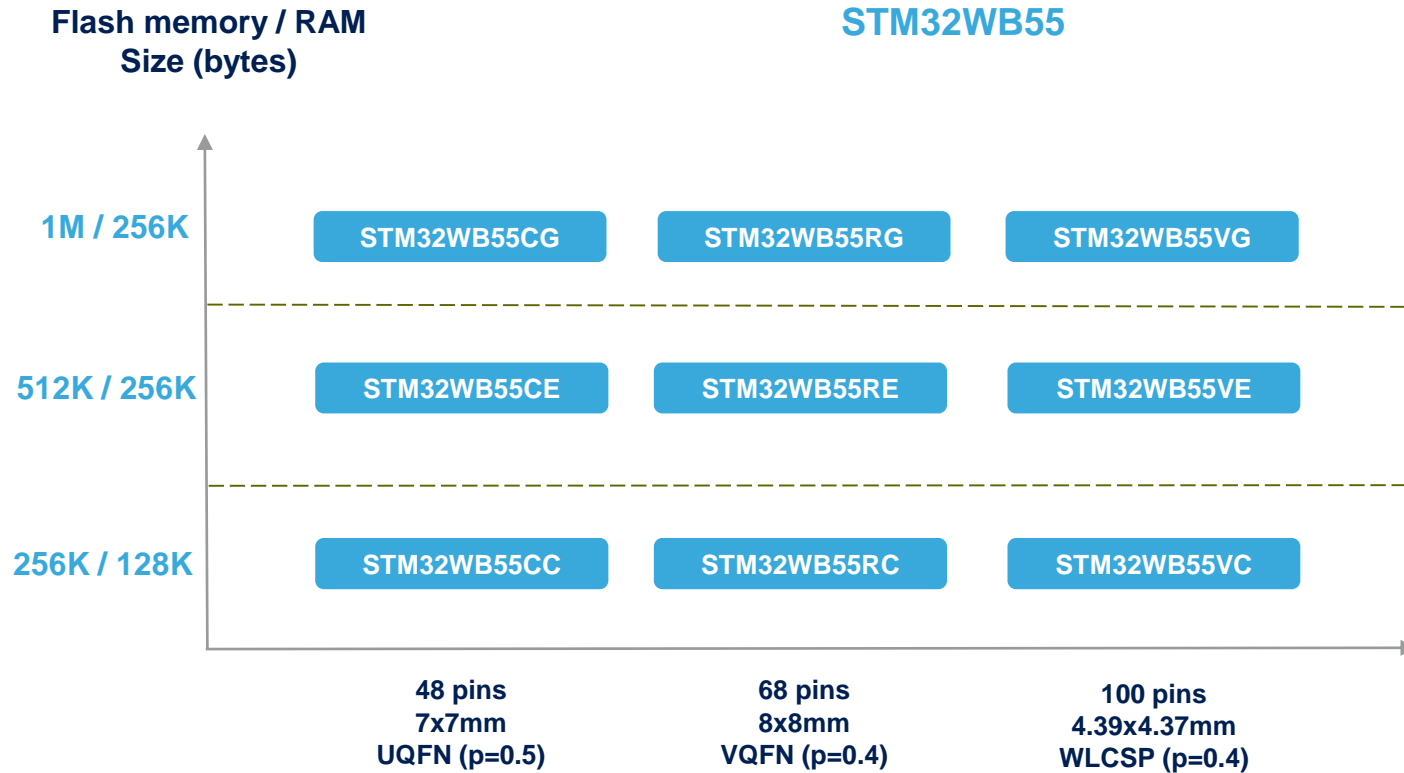
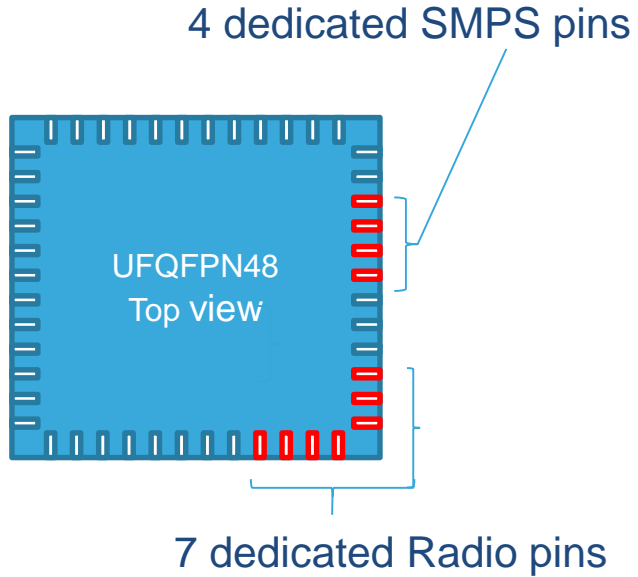


Specific STM32WB features 2/2

- 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

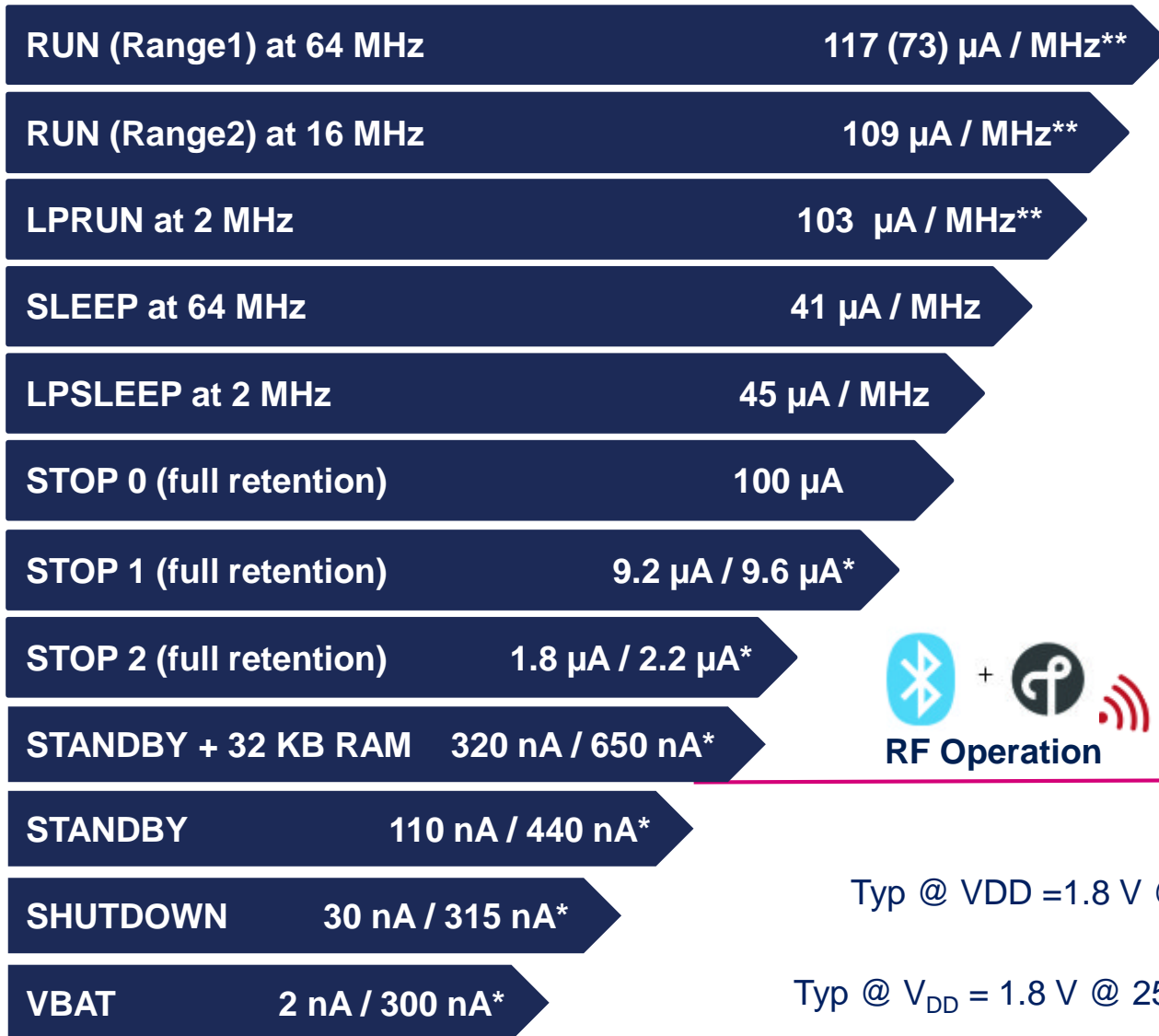




Low power modes

Wake-up time

9 cycles
9 cycles
1.7 μ s
4 μ s (19 μ s)
5 μ s (20 μ s)
14 μ s (25 μ s)
14 μ s (25 μ s)
50 μ s



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

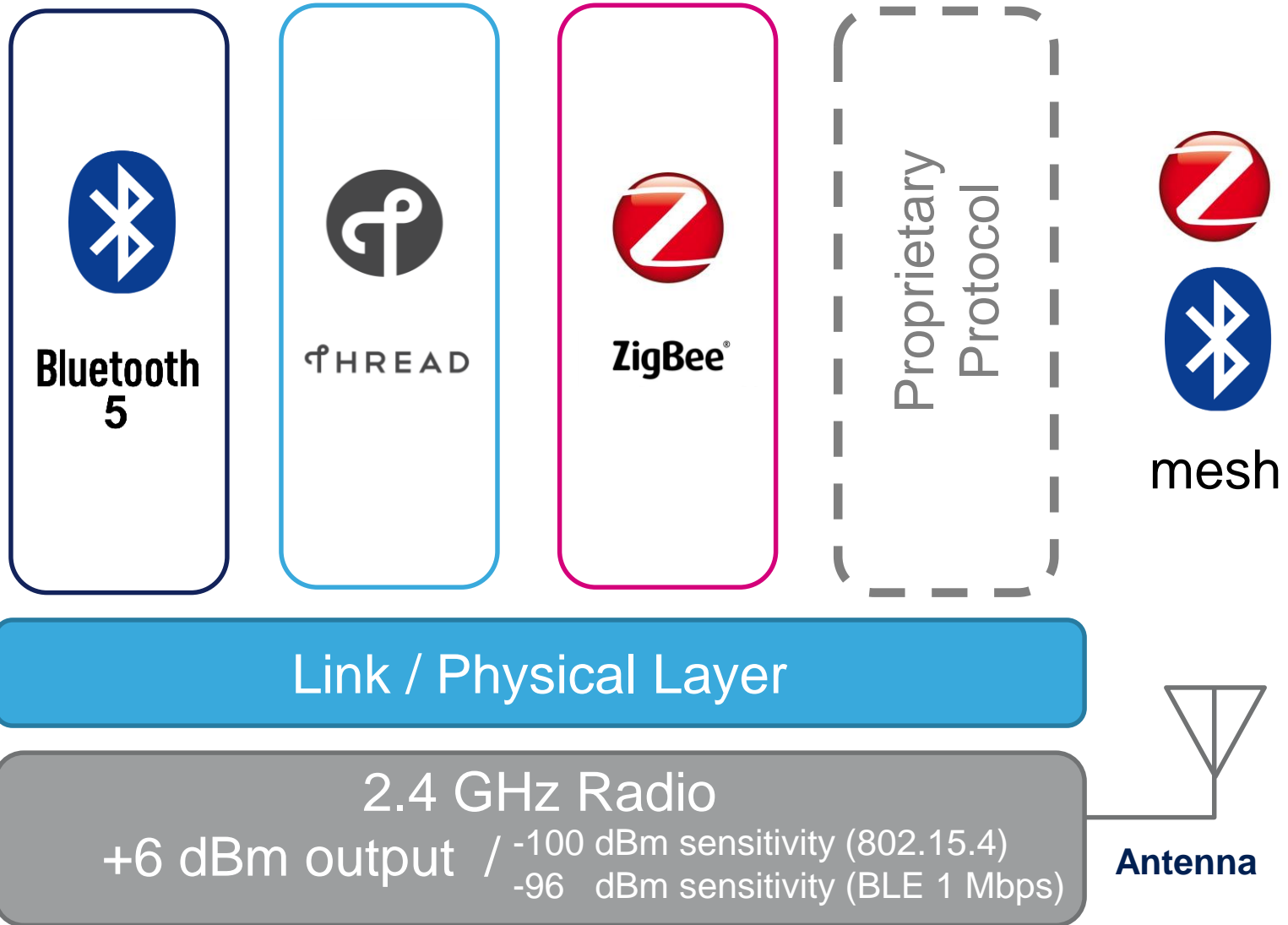
** : from SRAM1

() Typ. value for SMPS mode





Wireless part - make it Yours





- 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

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
- Cutable 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



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®
- IAR™ EWARM®
- 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



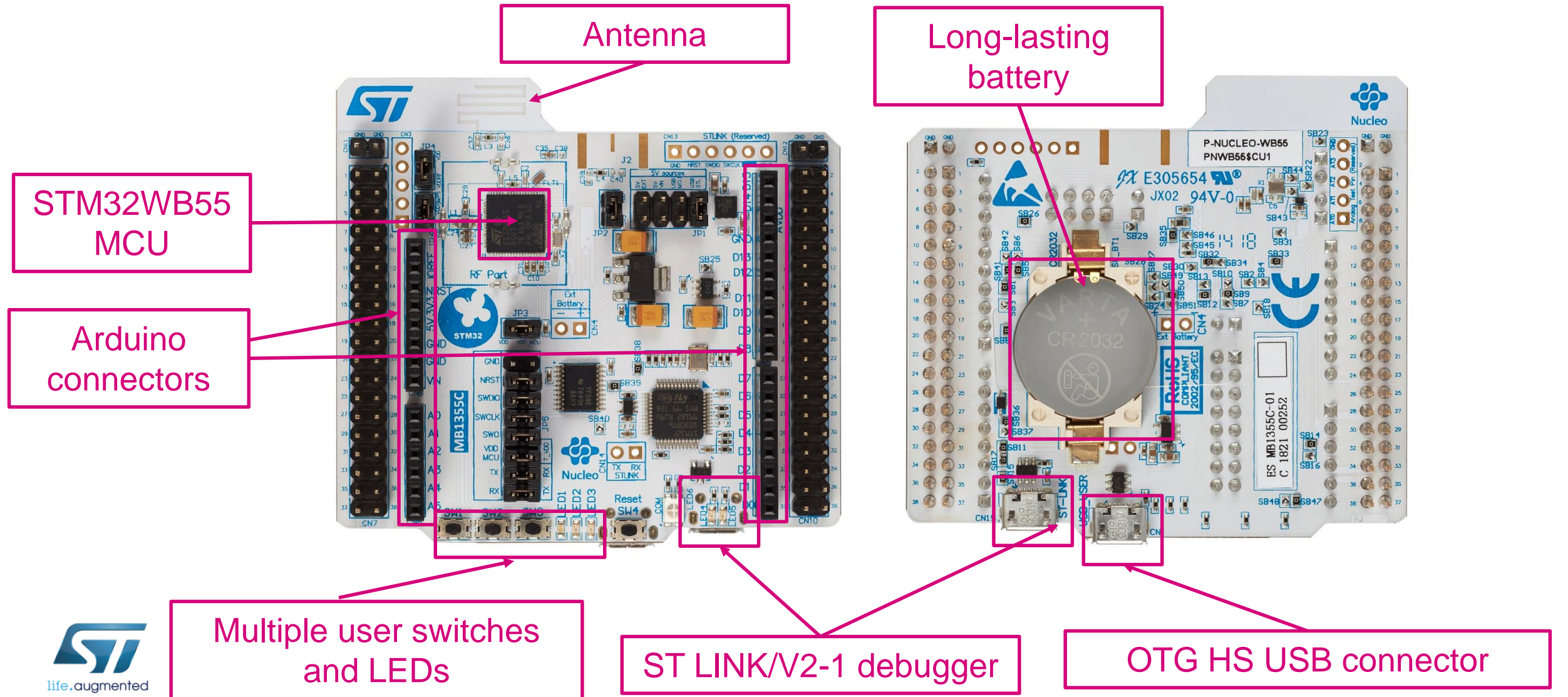
By using or installing (as applicable) this evaluation kit you accept all the terms of the EVALUATION PRODUCT LICENCE AGREEMENT available at www.st.com/epka

APPLICATIONS STORE

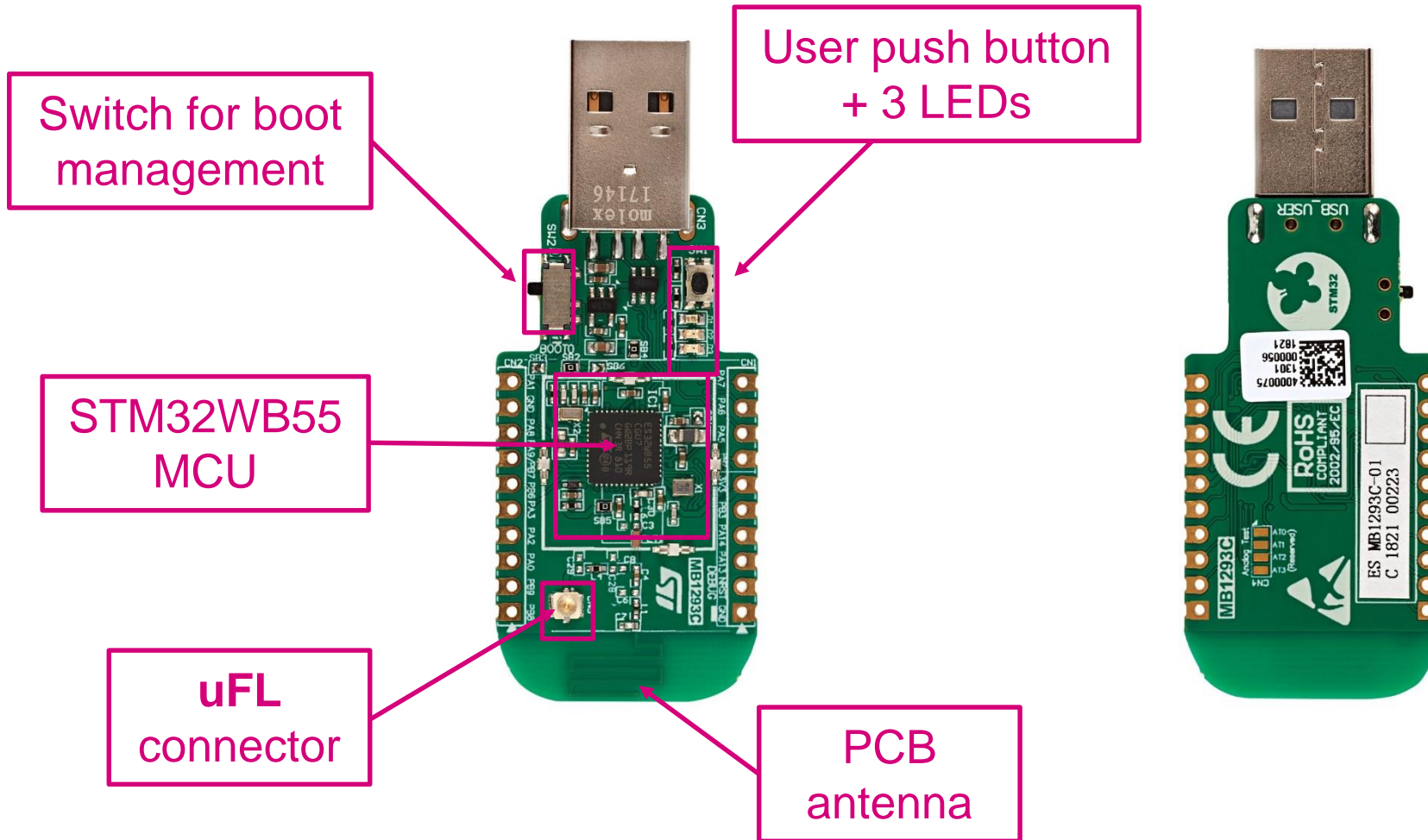


© STMicroelectronics - June 2018 - Printed in China - All rights reserved. The STMicroelectronics corporate logo is a registered trademark of the STMicroelectronics group of companies. All other names are the property of their respective owners.

STM32WB Nucleo Board



STM32WB Dongle



Key features

17

- 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.

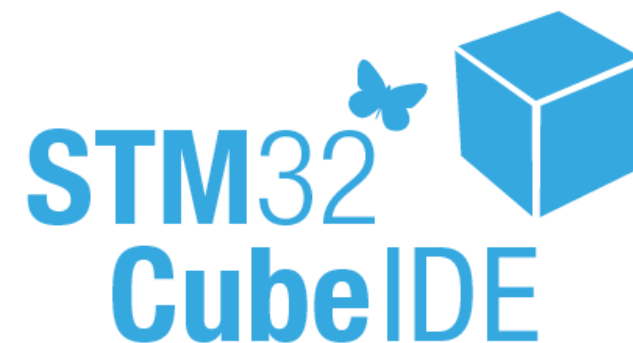
ARM®

Cortex™

Low-Power Leadership from ARM



- 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®



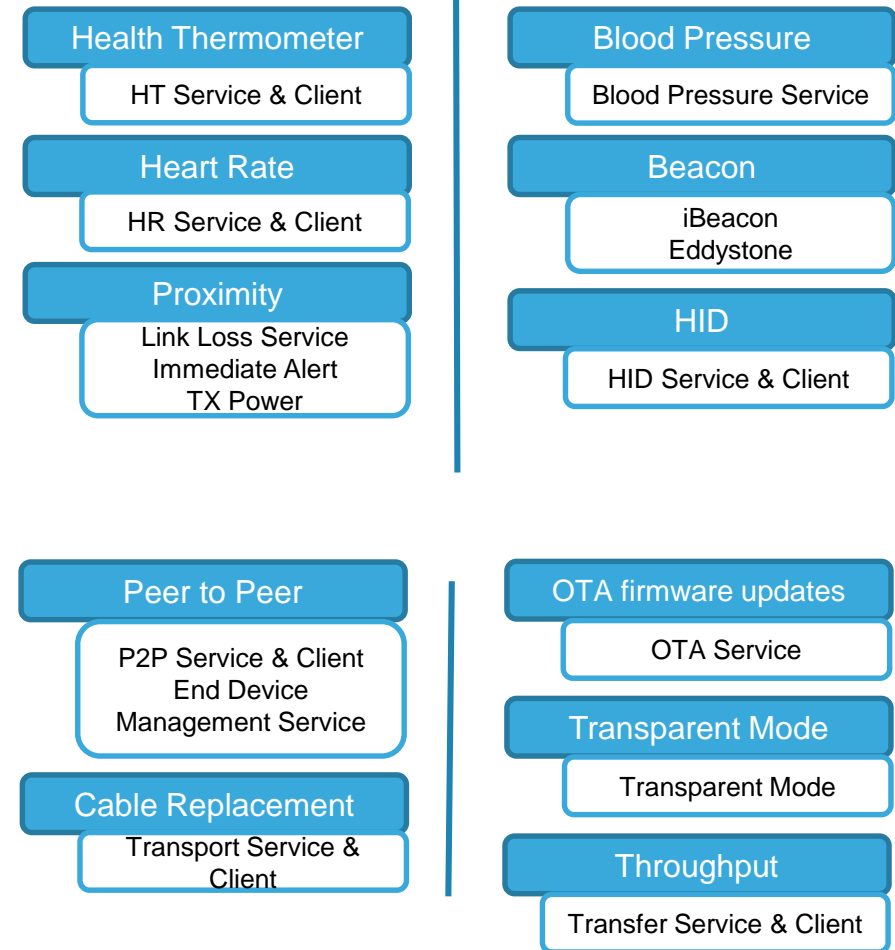
- This presentation summarizes the GATT-based applications developed for STM32WB and supported in the STM32Cube firmware library.

- **BT SIG GATT-based applications**

- Beacon
- Blood Pressure
- Health Thermometer
- Heart Rate
- Human Interface Device (HID)
- Proximity

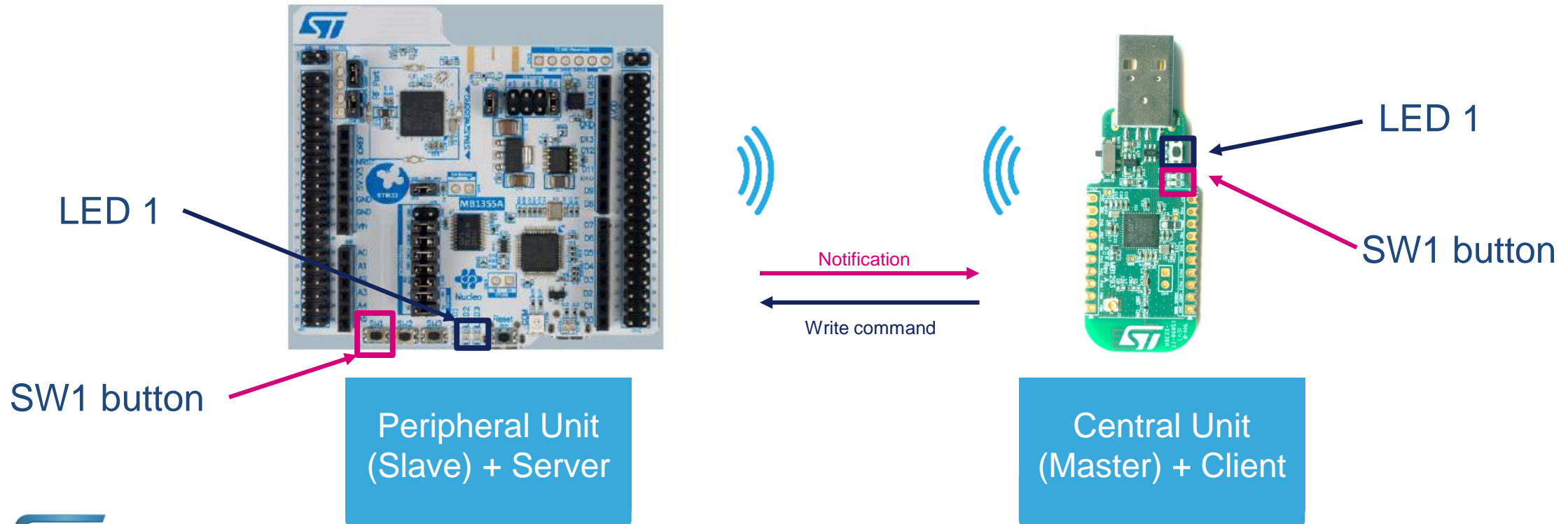
- **Proprietary GATT-based applications**

- 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

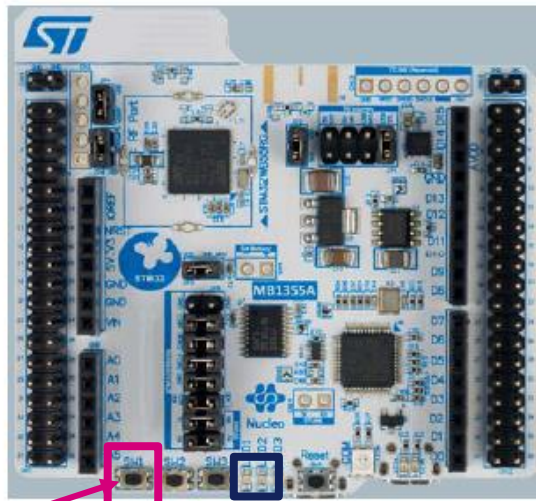


Note: The server and the client can be a Nucleo Board or a USB Dongle

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



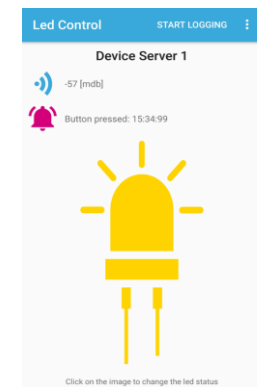
SW1 button

Peripheral Unit (Slave) + Server



Central Unit (Master) + Client

Use Your ST SensNet App



Virtual LED status

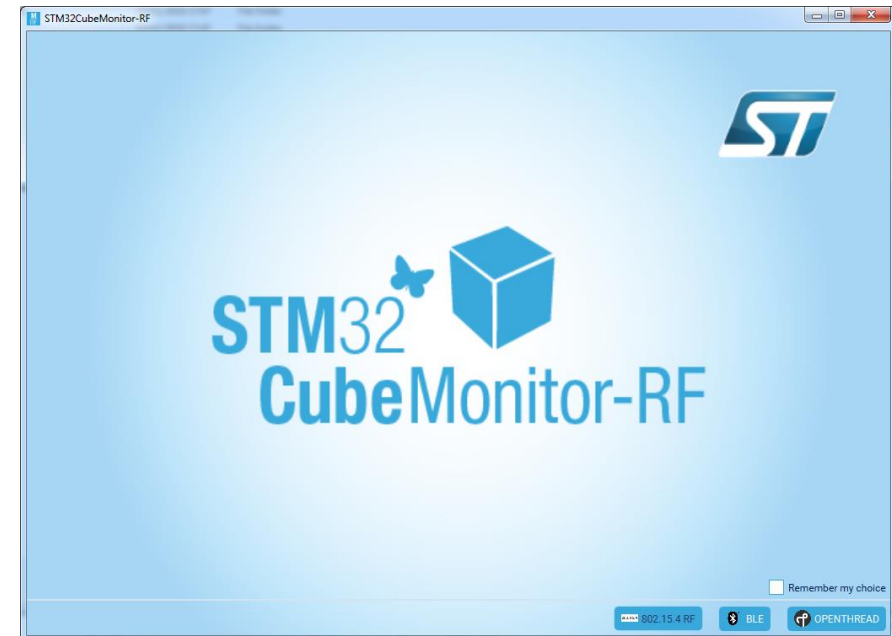
Note: The server can be a Nucleo Board or a USB Dongle

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



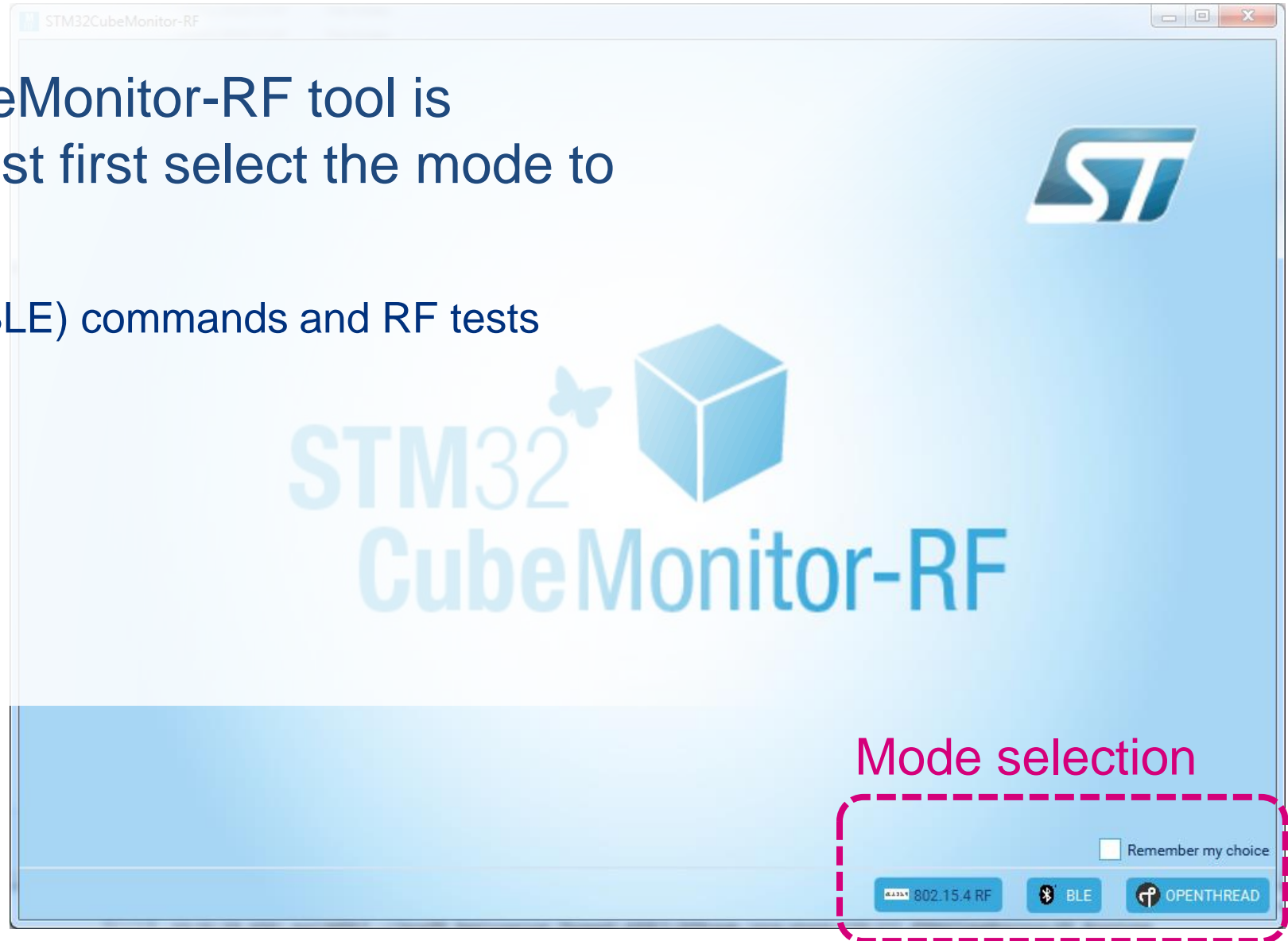
STM32CubeMonitor-RF

- 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.



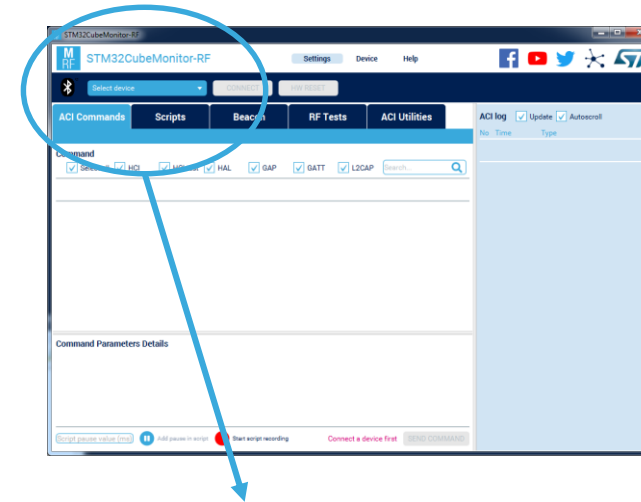
- With STM32CubeMonitor-RF in BLE mode, it is possible to:
 - Send ACI commands to the target board. The ACI commands are BLE commands used to configure and communicate with BLE peripherals. The tool provides the list of the available commands and parameter descriptions.
 - Perform RF tests (transmit,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



RF test with two devices

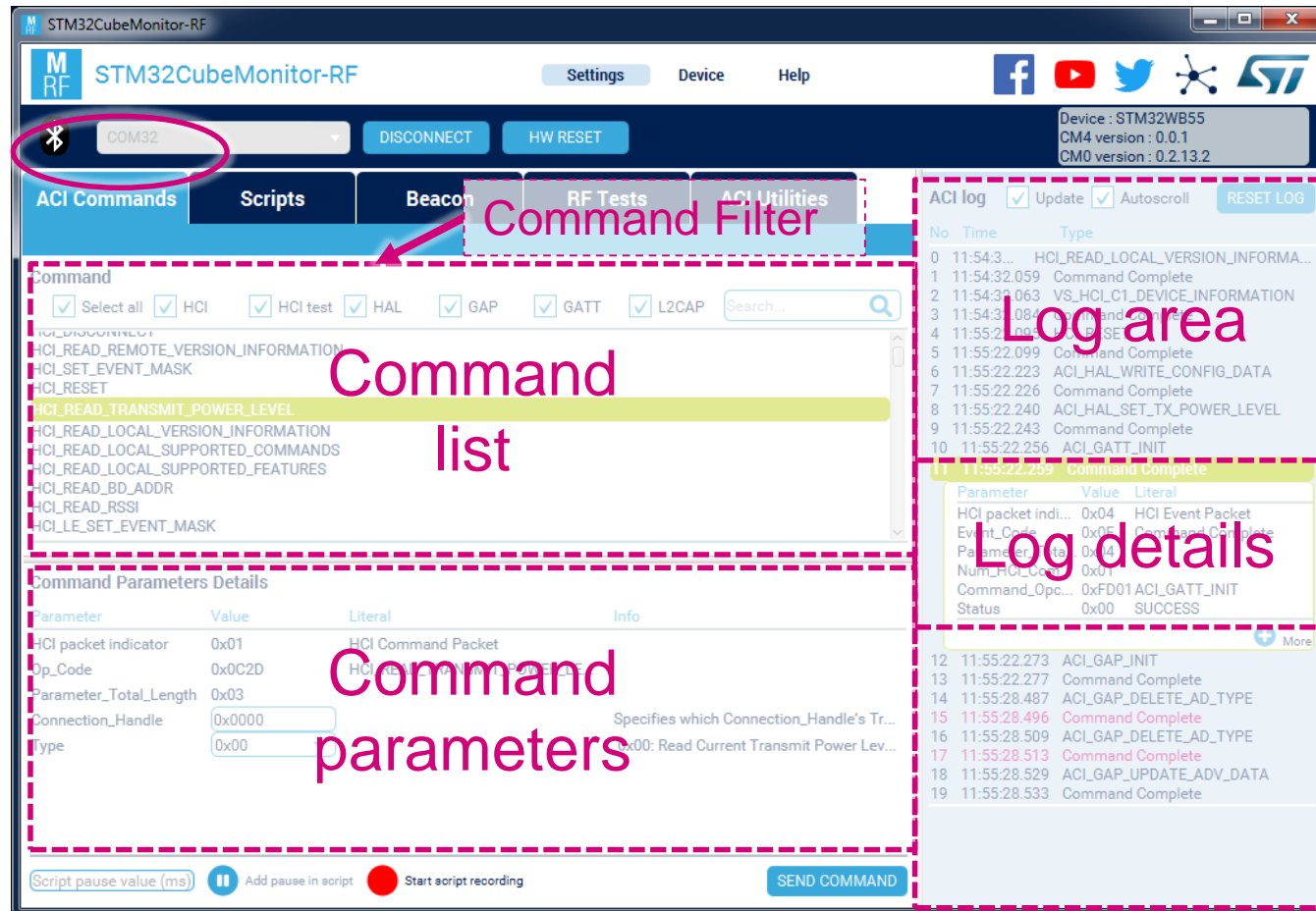
Connect the STM32WB device

- 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



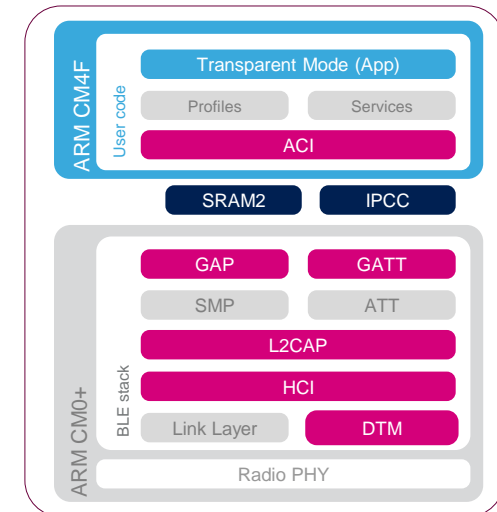
ACI command panel

- 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



Send one ACI command

28

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

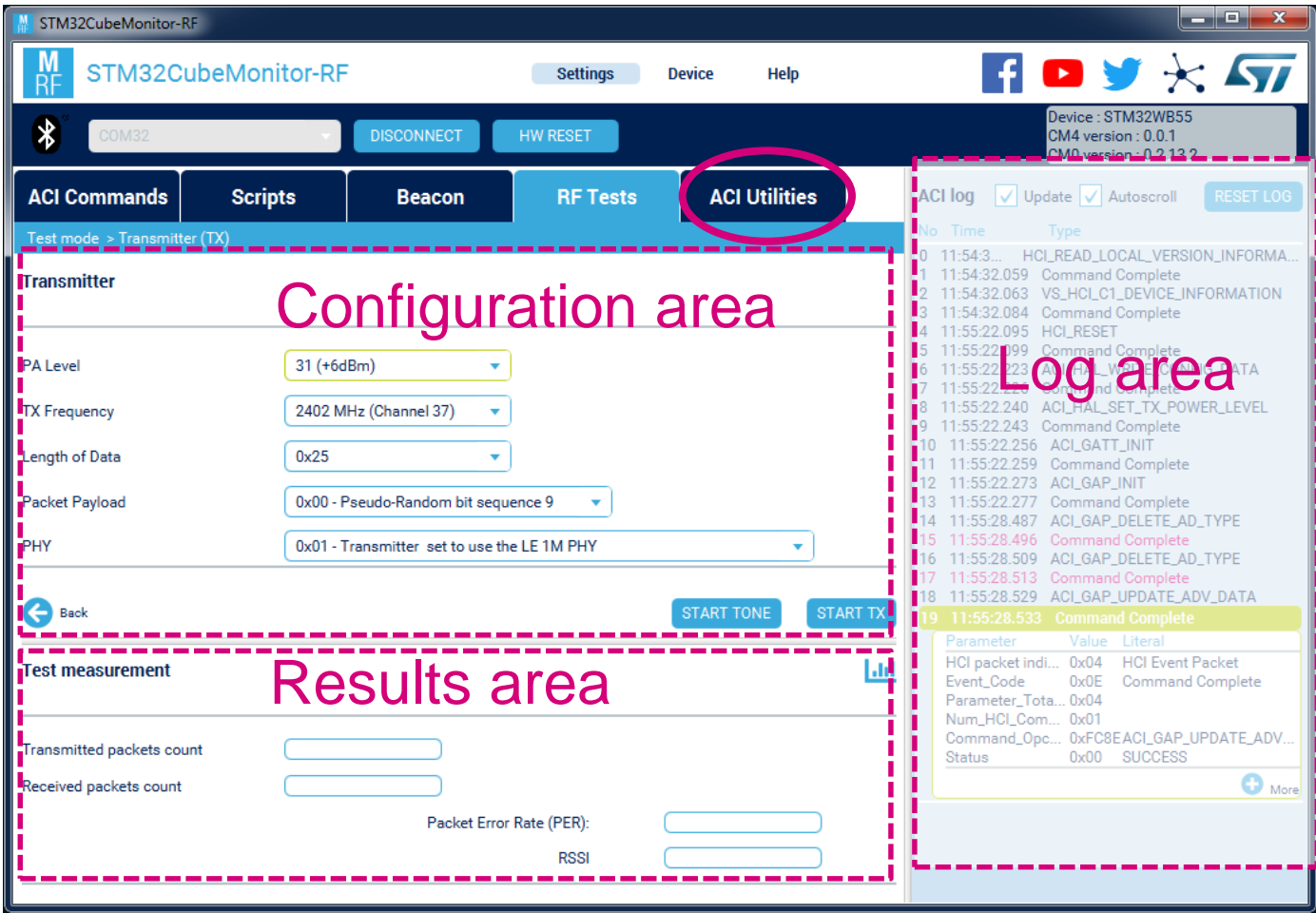
The screenshot shows the STM32CubeMonitor-RF application window. The interface includes a top menu bar with 'Settings', 'Device', and 'Help'. Below the menu is a device selection area with 'COM32' selected and buttons for 'DISCONNECT' and 'HW RESET'. The main area is divided into tabs: 'ACI Commands', 'Scripts', 'Beacon', 'RF Tests', and 'ACI Utilities'. The 'ACI Commands' tab is active, displaying a list of commands with checkboxes for 'Select all', 'HCI', 'HCI test', 'HAL', 'GAP', 'GATT', and 'L2CAP'. The command 'HCI_READ_TRANSMIT_POWER_LEVEL' is selected and highlighted in yellow, with a red circle '1' next to it. Below the command list is the 'Command Parameters Details' section, which shows a table of parameters for the selected command. The 'Connection_Handle' parameter is set to '0x0000' and the 'Type' parameter is set to '0x00'. A red circle '2' is next to the 'Type' dropdown. At the bottom right of the interface is a 'SEND COMMAND' button, with a red circle '3' next to it. On the right side of the window is the 'ACI log' area, which displays a list of log entries with columns for 'No', 'Time', and 'Type'. The log entry for the selected command is highlighted in yellow, and a pop-up window shows the details of the command, including 'Parameter', 'Value', and 'Literal'.

Parameter	Value	Literal	Info
HCI packet indicator	0x01	HCI Command Packet	
Op_Code	0x0C2D	HCI_READ_TRANSMIT_POWER_LE...	
Parameter_Total_Length	0x03		
Connection_Handle	0x0000		Specifies which Connection_Handle's Tr...
Type	0x00		0x00: Read Current Transmit Power Lev...

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

30

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

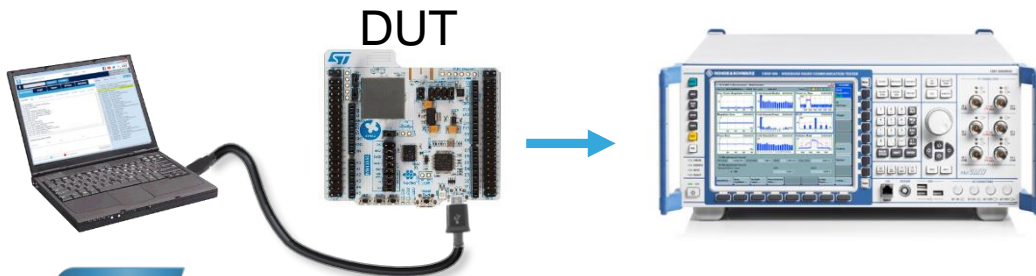


Test mode > Transmitter (TX)



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”



Test mode > Transmitter (TX)

Transmitter

PA Level: 25 (0dBm) 1

TX Frequency: 2410 MHz (Channel 3) 2

Length of Data: 0x7

Packet Payload: 0x00 - Pseudo-Random bit sequence 9

PHY: 0x01 - Transmitter set to use the LE 1M PHY

Back 3 **START TONE** **START TX** 4

Test measurement

Transmitted packets count:

Received packets count:

Packet Error Rate (PER):

RSSI:

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

Test mode > Transmitter (TX)

Transmitter

PA Level

TX Frequency 1

Length of Data

Packet Payload

PHY

2 3

Test measurement

Transmitted packets count 4

Received packets count

Packet Error Rate (PER):

RSSI

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

Test mode > Receiver (RX)

Receiver

RX Frequency: 2402 MHz (Channel 37) 1

PHY: 0x01 - Receiver set to use the LE 1M PHY 2

Index modulation: 0x00 - Assume transmitter will have a standard modulation index 3

Get RSSI

[← Back](#) 4 5 [START RX](#)

Test measurement

Transmitted packets count:

Received packets count: 6

Packet Error Rate (PER):

RSSI:

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”

Test mode > Packet Error Rate (PER)

Connect tester

COM37 3 DISCONNECT

Device : STM32WB55
CM4 version : 0.0.1
CM0 version : 0.2.13.2

Disconnect 2nd device to change test mode 4 CONFIGURE TESTER

Packet Error Rate test: Step 2

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

Test mode > Packet Error Rate (PER) > COM37

Configure tester (COM37)

PA Level	<input type="text" value="22 (-0.85dBm)"/>	
TX Frequency	<input type="text" value="2410 MHz (Channel 3)"/>	5
Length of Data	<input type="text" value="0x25"/>	
Packet Payload	<input type="text" value="0x00 - Pseudo-Random bit sequence 9"/>	
PHY	<input type="text" value="0x01 - Transmitter set to use the LE 1M PHY"/>	

6

Packet Error Rate test: Step 3

37

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

Test mode > Packet Error Rate (PER) > COM37 > COM32

Configure Device under test (DUT) (COM32)

RX Frequency

2402 MHz (Channel 37) ▼

7

PHY

0x01 - Receiver set to use the LE 1M PHY ▼

Index modulation

0x00 - Assume transmitter will have a standard modulation index ▼



8

CONFIGURE PARAM

Packet Error Rate test: Step 4

38

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.

Test mode > Packet Error Rate (PER) > COM37 > COM32 > Settings

Configure additional settings

PER tests on multiple channels Fill channel List:

Get RSSI 9 Measurement period (sec):

Save test verdict in file

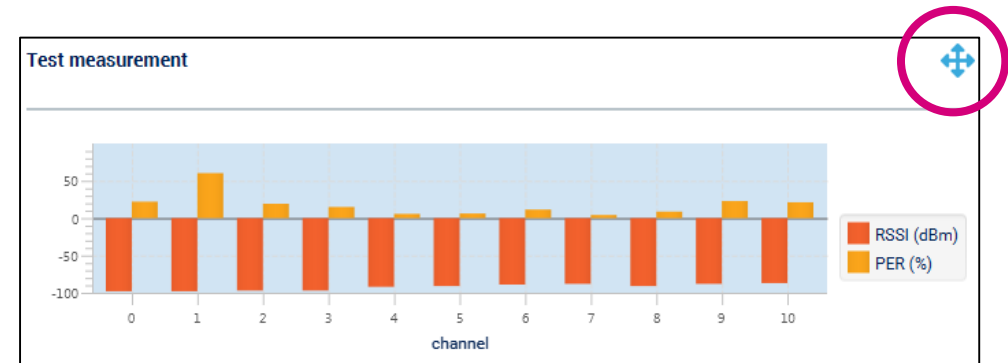
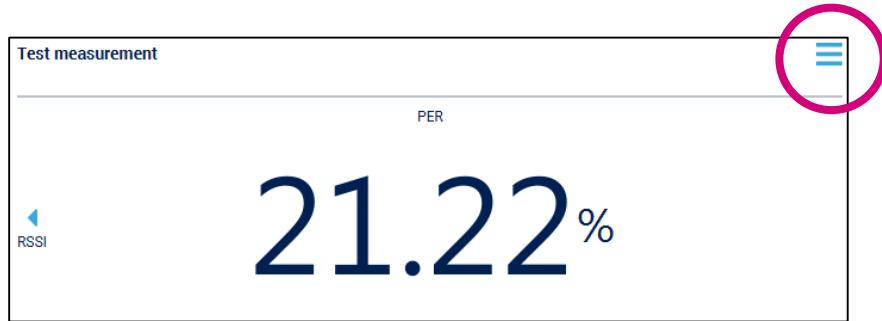
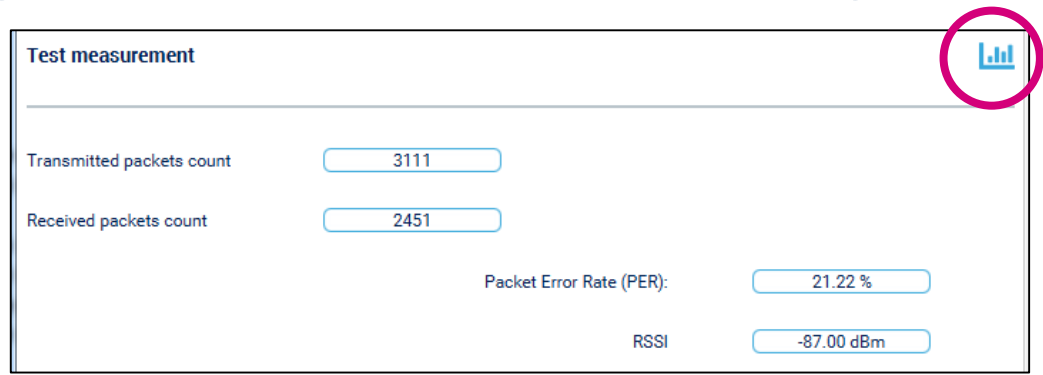
10

Test measurement

Transmitted packet number	<input type="text" value="3089"/>
Received packet number	<input type="text" value="3089"/> 11
Packet Error Rate (PER):	<input type="text" value="0.00 %"/>
RSSI	<input type="text" value="-43.00 dBm"/>

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 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

The screenshot shows the 'Scripts' tab in the ACI interface. At the top, there are navigation tabs: 'ACI Commands', 'Scripts', 'Beacon', 'RF Tests', and 'ACI Utilities'. Below the tabs, the 'Script' section is active. A checkbox labeled 'Generate report' is checked. A text input field contains the file path 'C:\Data\MTT\BLE\MonitorRF\SampleScript.txt', with a 'BROWSE' button to its right. The main area displays the script's content, which includes comments and commands: '# 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;0x00)', '# Wait 3 seconds Wait(3000)', '# Send stop tone Send (ACI_HAL_TONE_STOP)', and '# Pause command Pause("End of script")'. A 'START SCRIPT' button is located at the bottom right of the interface.

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

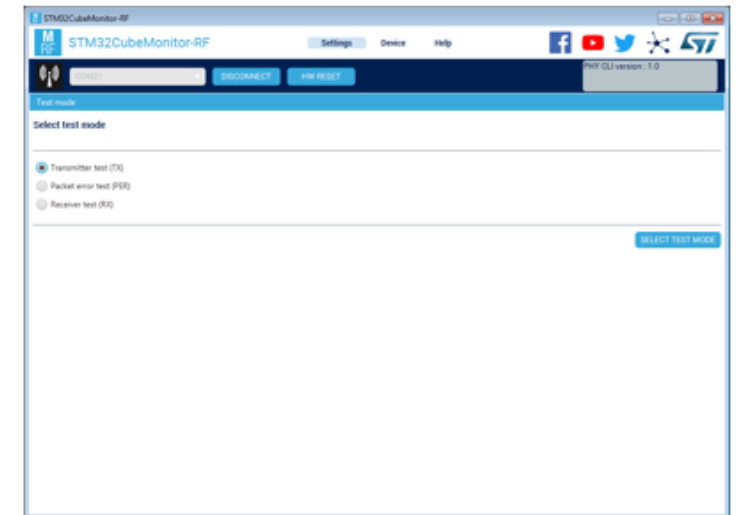
42

- Thread is an IoT protocol for connected devices in buildings and homes.
- STM32WB devices support the OpenThread stack that implements the Thread protocol



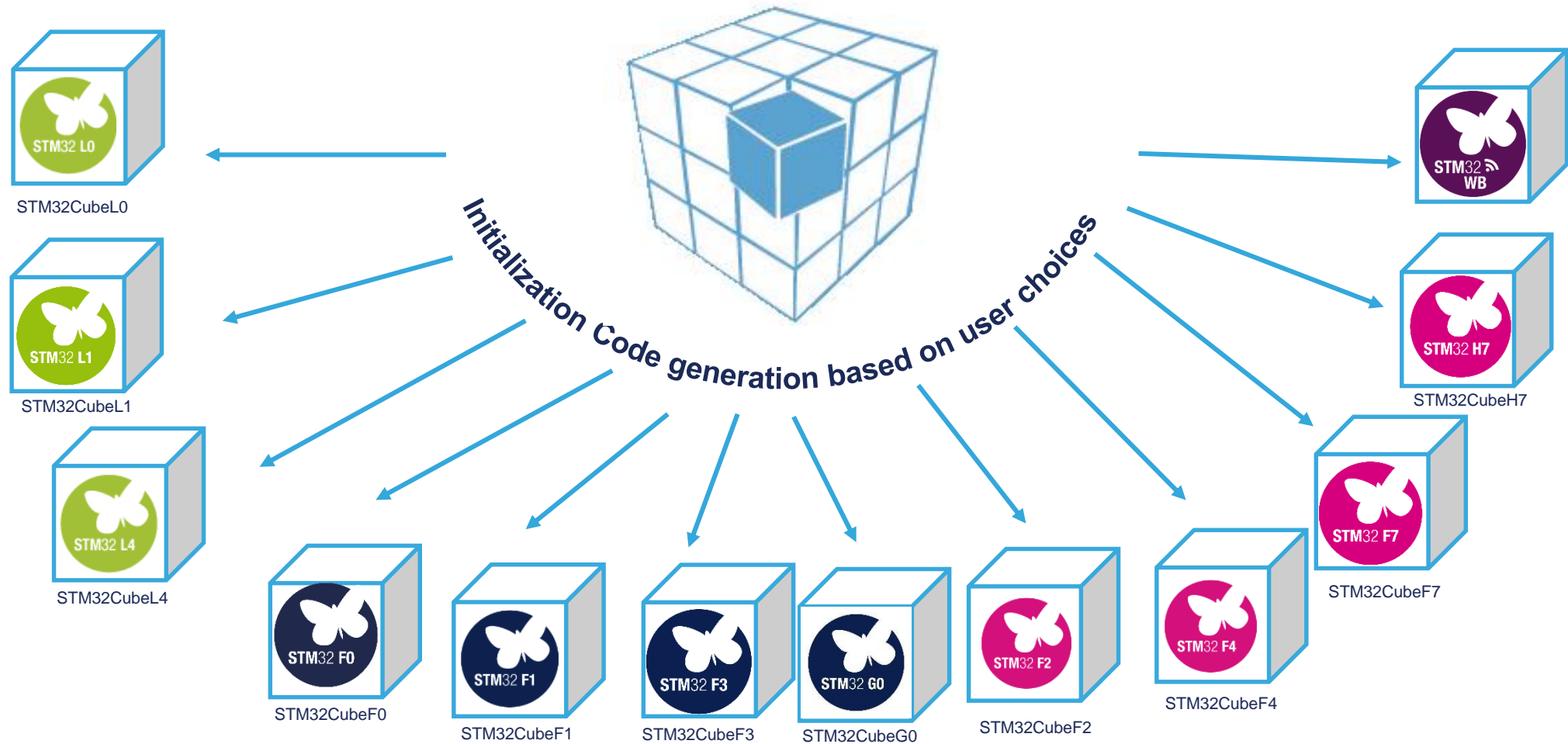
- The OT (OpenThread) command line API is used to send commands to the stack (CLI).

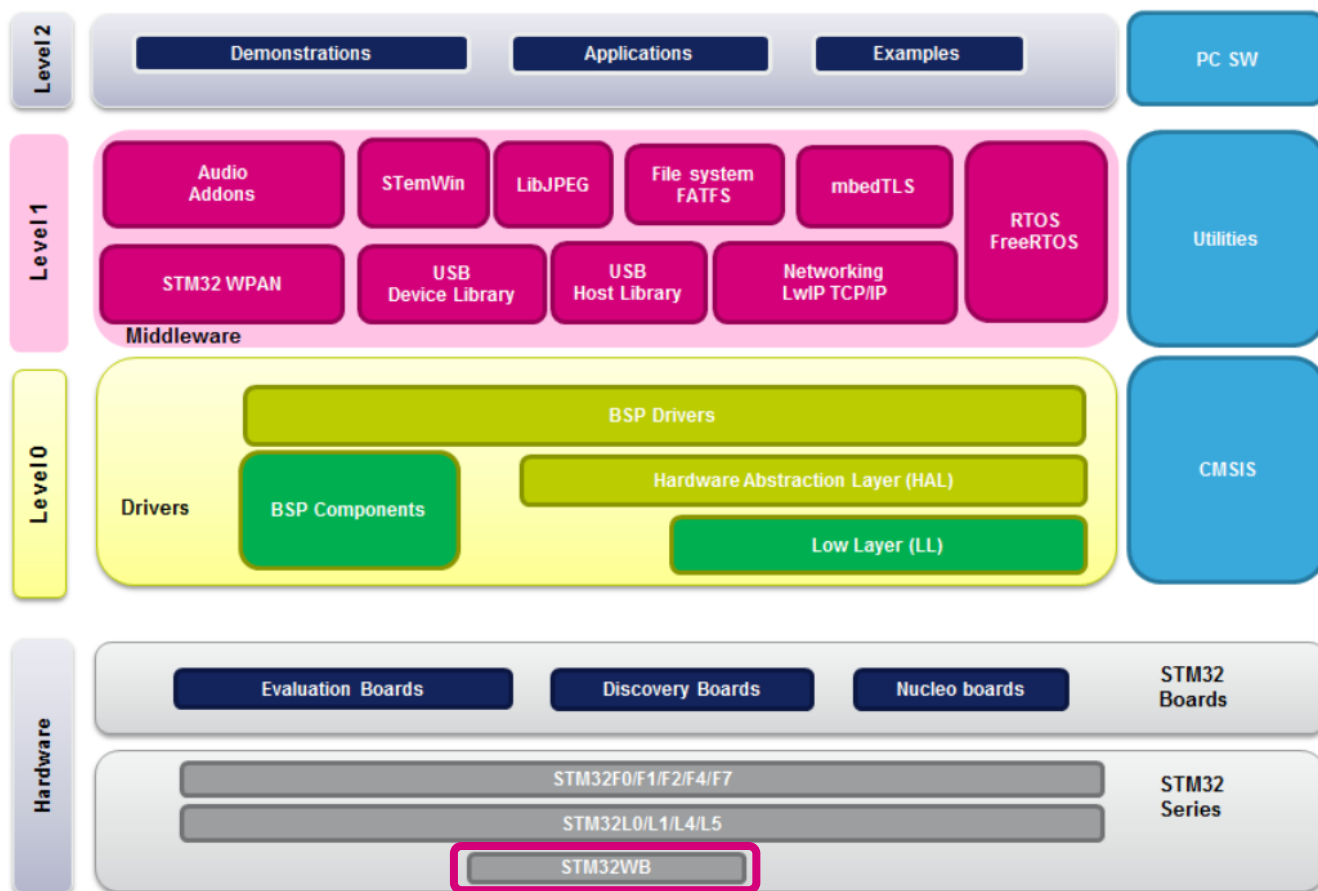
- IEEE 802.15.4 RF mode is used to test IEEE 802.15.4 devices.
- Thread and Zigbee use IEEE 802.15.4
- The tool allows to:
 - Test transmission,
 - Test reception
 - Measure PER



Development environment

STM32CubeMX





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!

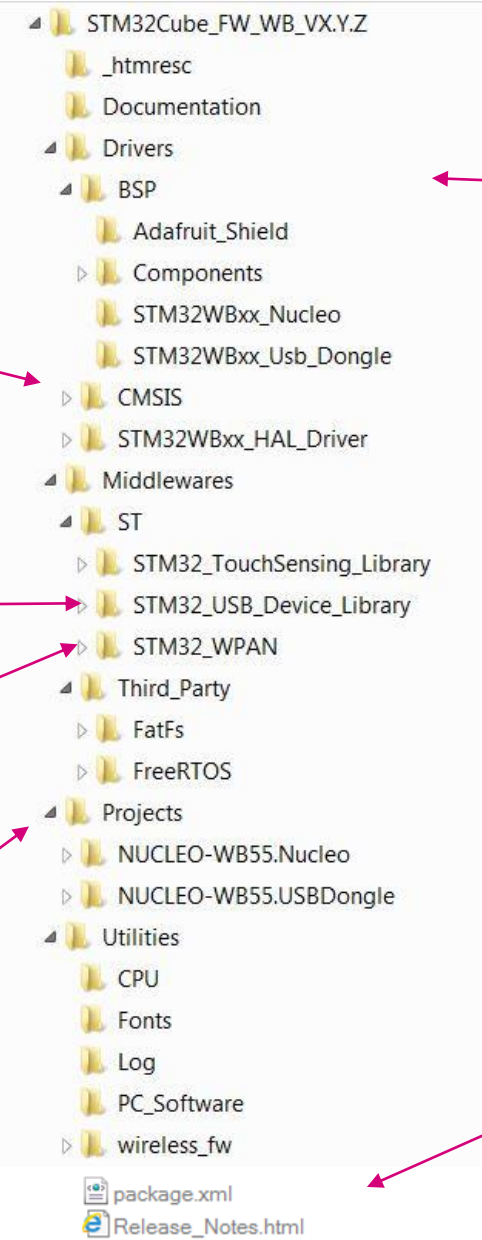
Package organization

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

Touch Sensing library

Wireless Personal Area Network library

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



BSP drivers for the supported boards

STM32WBxx HAL drivers for all modules

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

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

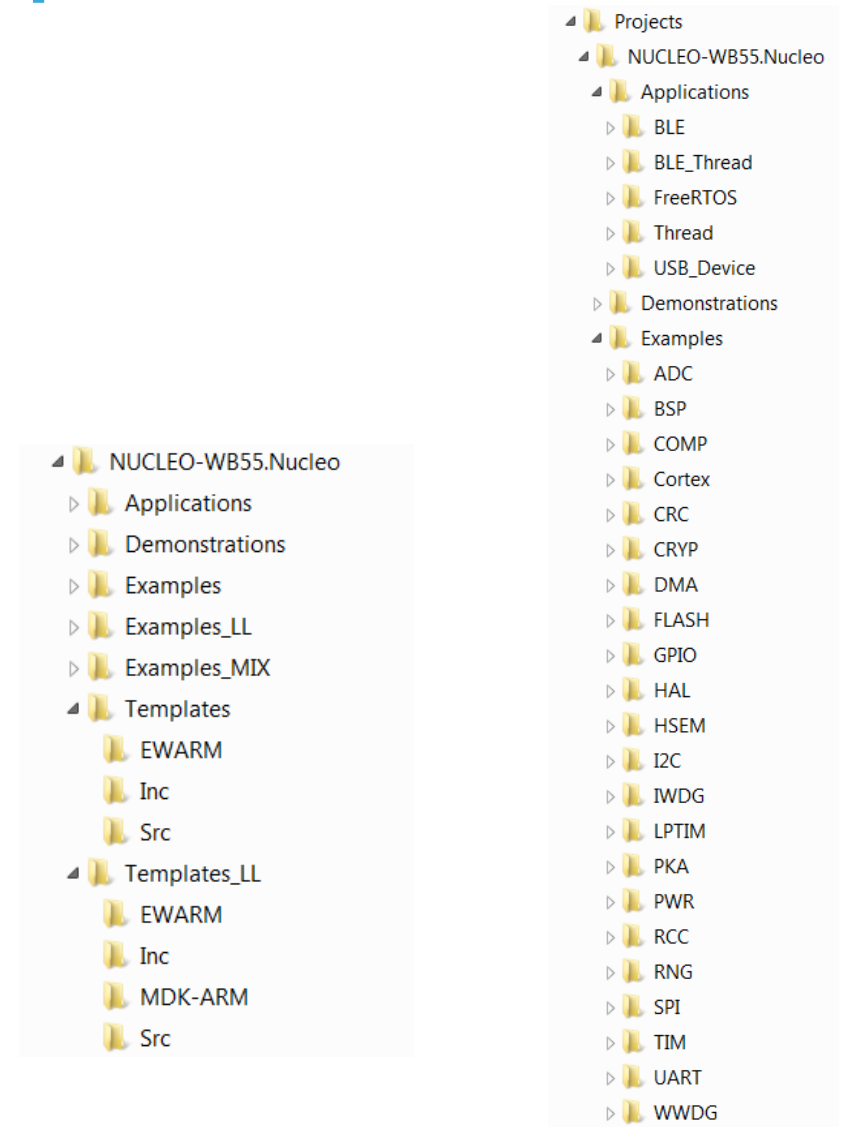
Board	Example	Application	Demonstration
STM32WB55 Nucleo pack	88	39	22

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.



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](#)

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

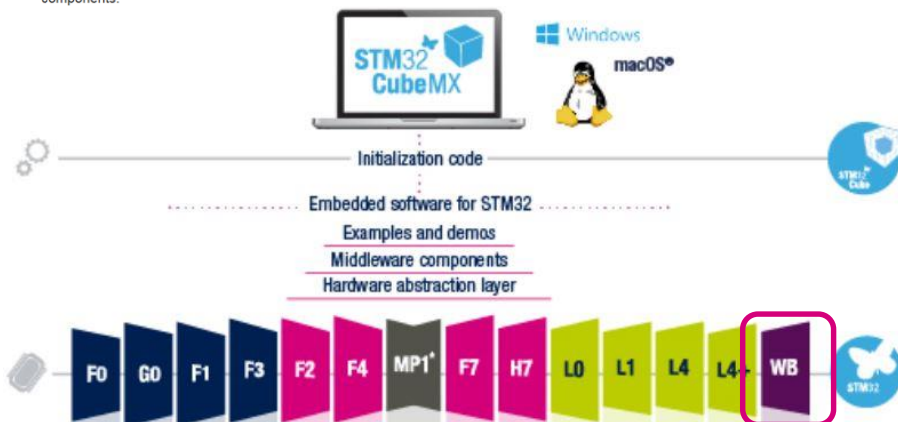
STM32Cube MCU & MPU Packages

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.



macOS® is a trademark of Apple Inc., registered in the U.S. and other countries.
Note: * available for Cortex-M4 side only



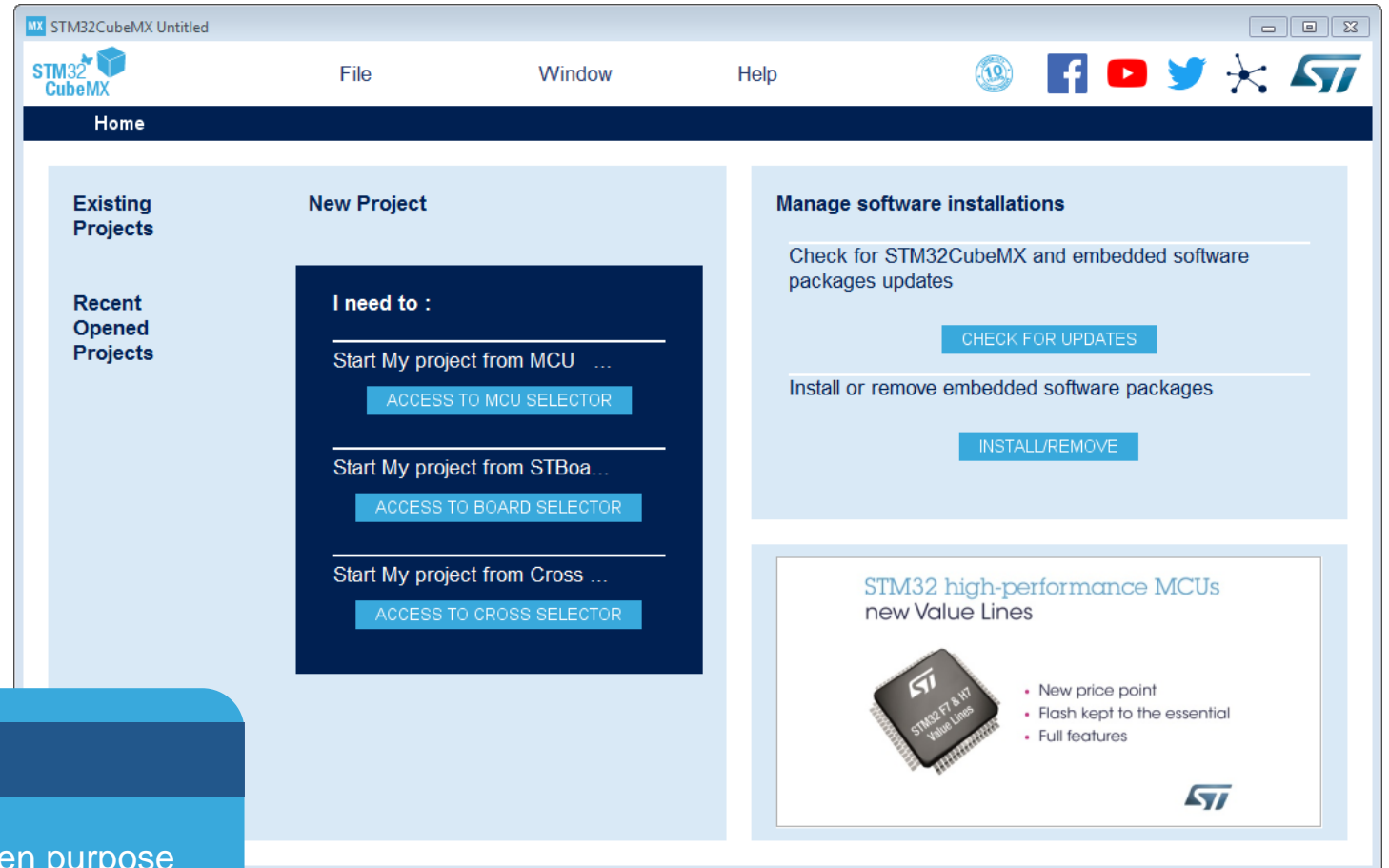
Part Number	Supplier	Supported Devices	STM32CubeMX Compatible
<input type="text" value="Part Filter"/> <input type="button" value="Q"/>			
▶ STM32CubeH7 STM32Cube MCU Package for STM32H7 series (HAL low level drivers, USB, TCP/IP, File system, RTOS)...	ST	STM32H7	true
▶ STM32CubeMP1 STM32CubeMP1 Package 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

- Choose ideal MCU and simply configure

- Pinouts
- Clocks and oscillators
- Peripherals
- Low-power modes
- Middleware

Application benefits

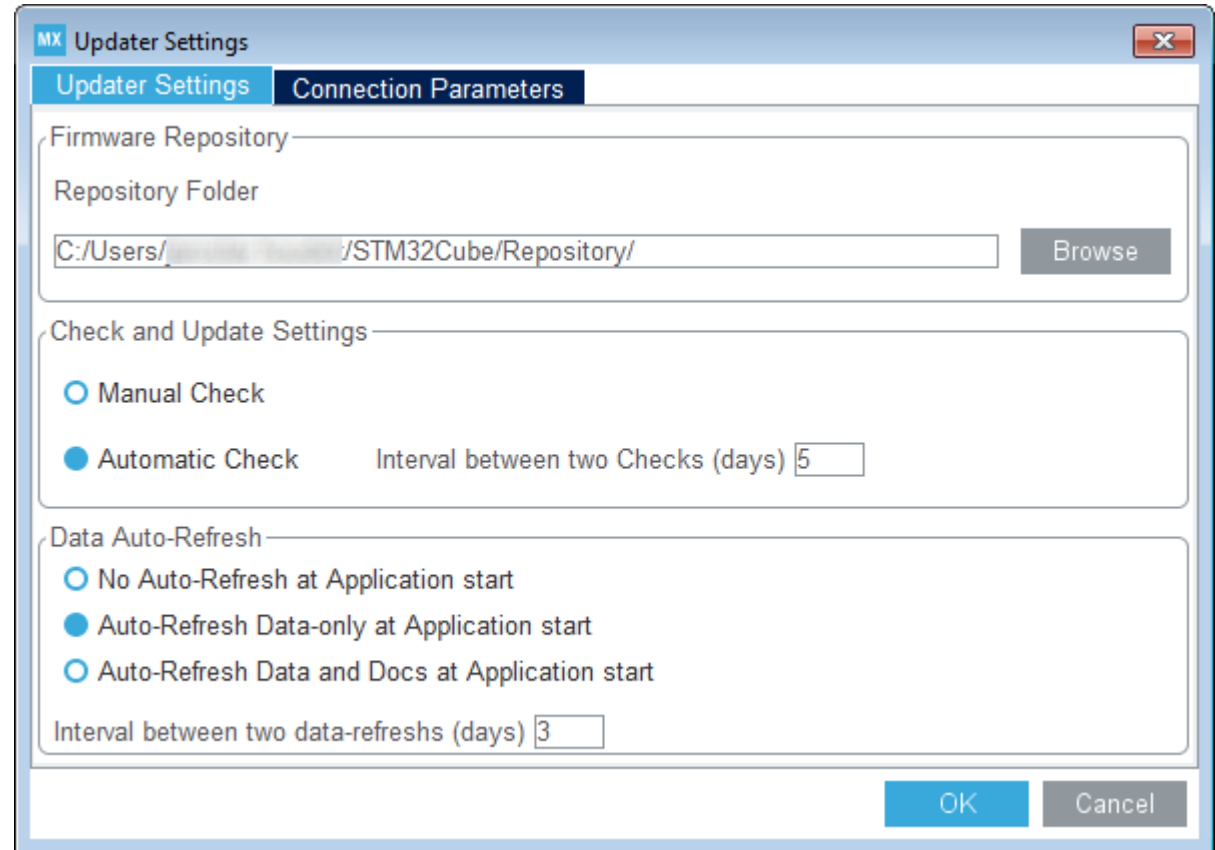
- Helps choose the correct MCU for a given purpose
- Simulation provides an advantage in design phase
- Boosts development speed with a headstart



- 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.

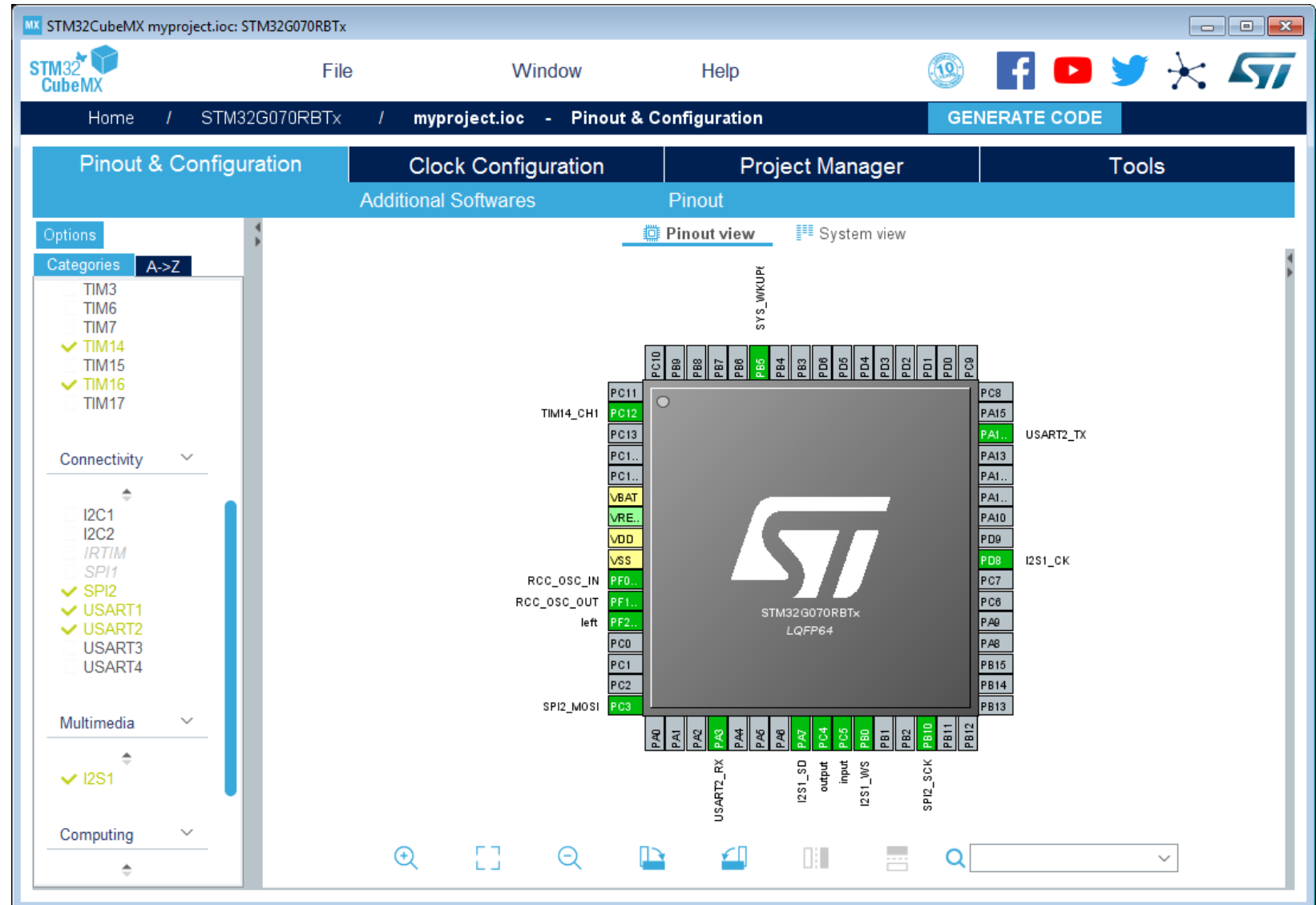
Prerequisites and settings

- 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.

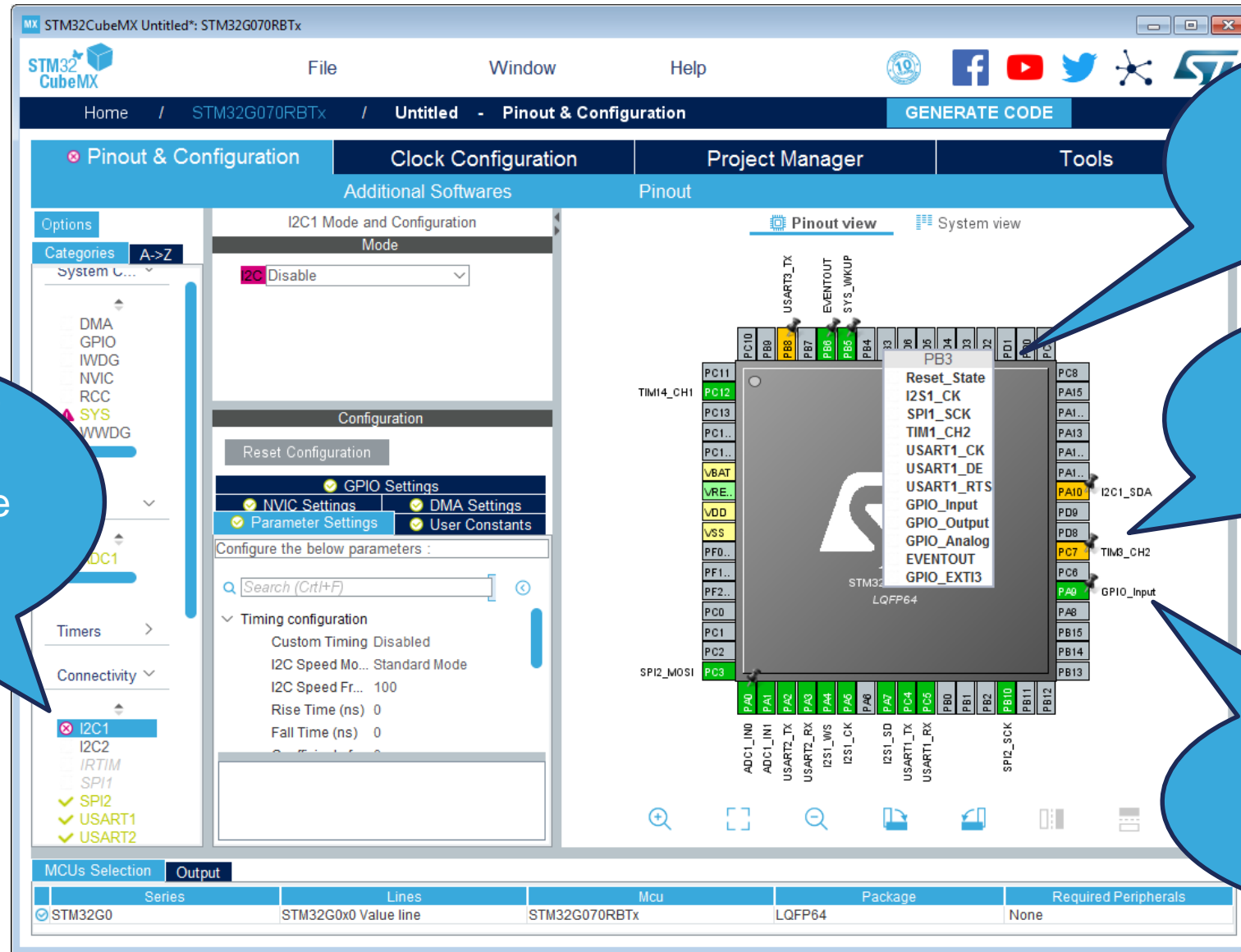


- Find MCU by name ...
 - Quickly locate by Series and Lines
- ... or application needs
 - Package (pin count)
 - RAM size
 - NV memory requirements
 - Embedded peripherals
 - Number and type of interfaces
 - Core and frequency
 - Price
- Convenient links to documentation
- Export table to Excel file

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



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
 - ✓ OK
 - ⚠ Non-blocking problem
 - ✗ Error

Click to configure DMA

Quickly switch between pinout and system views

GPIO configuration is considered incorrect, but code may be generated

Error in configuration, code generator will display a warning message.

Configuration is valid here

System Core	Analog	Timers	Connectivity	Multimedia	Computing
DMA ⚠	ADC1 ✓	TIM14 ✓	SPI2 ✓	I2S1 ✓	
GPIO ✗		TIM15 ✗	USART1 ✓		
NVIC ✓		TIM16 ✓	USART2 ✓		
RCC ✓					

MCUs Selection	Series	Package	Required Peripherals
STM32G0	STM32G0	LQFP64	None

Middleware configuration

59

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

The screenshot displays the STM32CubeMX software interface for configuring the FATFS mode and configuration. The main window is titled "STM32CubeMX Untitled*: STM32G070RBTx". The interface includes a menu bar (File, Window, Help), a breadcrumb trail (Home / STM32G070RBTx / Untitled - Pinout & Configuration), and a "GENERATE CODE" button. The main content area is divided into several sections: "Pinout & Configuration", "Clock Configuration", "Project Manager", and "Tools". The "Pinout & Configuration" section is active, showing "Additional Softwares" and "Pinout" options. The "Options" section is expanded to show "Categories" (A->Z) and "Options" (USART1, USART2, USART3, USART4). The "Multimedia" section is expanded to show "I2S1". The "Computing" section is expanded to show "CRC". The "Middleware" section is expanded to show "FATFS" and "FREERTOS". The "FATFS Mode and Configuration" section is expanded to show "Mode" (User-defined) and "Configuration" (Reset Configuration, Set Defines, Advanced settings, User Constants). The "Configure the below parameters" section is expanded to show a search bar and a list of parameters:

Parameter	Value
FATFS version	R0.12c
FS_READONLY (Read-only mode)	Disabled
FS_MINIMIZE (Minimization level)	Disabled
USE_STRFUNC (String functions)	Enabled with LF -> CRLF conversion
USE_FIND (Find functions)	Disabled
USE_MKFS (Make filesystem function)	Enabled
USE_FASTSEEK (Fast seek function)	Enabled
USE_EXPAND (Use f_expand function)	Disabled
USE_CHMOD (Change attributes function)	Disabled
USE_LABEL (Volume label functions)	Disabled
USE_FORWARD (Forward function)	Disabled

The "MCUs Selection" section is expanded to show "Output" and a table of MCU options:

Series	Lines	Mcu	Package	Required Peripherals
STM32G0	STM32G0x0 Value line	STM32G070RBTx	LQFP64	None

Peripheral configuration

60

- 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.

The screenshot displays the STM32CubeMX interface for configuring a USART2 peripheral. The main window is titled "STM32CubeMX Untitled*: STM32G070RBTx". The top menu bar includes "File", "Window", and "Help". The breadcrumb navigation shows "Home / STM32G070RBTx / Untitled - Pinout & Configuration". A "GENERATE CODE" button is visible in the top right. The interface is divided into several sections: "Pinout & Configuration", "Clock Configuration", "Project Manager", and "Tools". Under "Pinout & Configuration", there are sub-sections for "Additional Softwares" and "Pinout". The "Options" section on the left includes "Categories" (A-Z), "Timers", "Connectivity", "Multimedia", and "Computing". The "Connectivity" section is expanded, showing a list of peripherals: I2C1, I2C2, IRTIM, SPI1, SPI2, USART1, USART2 (selected), USART3, and USART4. The "Multimedia" section shows I2S1 selected. The "Computing" section shows CRC selected. The main configuration area is titled "USART2 Mode and Configuration". It has a "Mode" section with a dropdown menu set to "Multiprocessor Communication". Below this are "Hardware Flow Control (RS232)" set to "Disable", "Hardware Flow Control (RS485)" (unchecked), and "Slave Select(NSS) Management" set to "Disable". The "Configuration" section has a "Reset Configuration" button and a "Parameter Settings" tab selected. Below the tabs are "User Constants", "NVIC Settings", "DMA Settings", and "GPIO Settings". A search bar is present with the text "Search (Ctrl+F)". The "Basic Parameters" section includes: Baud Rate (115200 Bits/s), Word Length (7 Bits (including Parity)), Parity (None), and Stop Bits (1). The "Advanced Parameters" section includes: Data Direction (Receive and Transmit), Over Sampling (16 Samples), Single Sample (Disable), Wake-Up Method (Idle Line), and ClockPrescaler (clock /1). At the bottom, there is a table for "MCUs Selection" and "Output".

Series	Lines	Mcu	Package	Required Peripherals
STM32G0	STM32G0x0 Value line	STM32G070RBTx	LQFP64	None

NVIC configuration panel

61

- 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.

The screenshot shows the STM32CubeMX software interface for configuring the NVIC. The main window is titled "STM32CubeMX Untitled*: STM32G070RBTx". The "Pinout & Configuration" tab is active, and the "NVIC Mode and Configuration" section is expanded to show the "Code generation" mode. The "Configuration" section includes a search bar and checkboxes for "Sort by Preemption Priority and Sub Priority" and "Show only enabled interrupts". The "NVIC Interrupt Table" is displayed as follows:

Interrupt	Enabled	Preemption Priority	Uses FreeRTOS...
Non maskable interrupt	<input checked="" type="checkbox"/>	0	<input type="checkbox"/>
Hard fault interrupt	<input checked="" type="checkbox"/>	0	<input type="checkbox"/>
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	<input type="checkbox"/>
Pendable request for system service	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
Time base: System tick timer	<input checked="" type="checkbox"/>	3	<input type="checkbox"/>
PVD interrupt through EXTI line 16	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
Flash global interrupt	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
RCC global interrupt	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
ADC1 interrupt	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
TIM14 global interrupt	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
TIM16 global interrupt	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
SPI1 global interrupt	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
SPI2 global interrupt	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>
USART2 global interrupt / USART2 wake-up interrupt through EXTI line 26	<input type="checkbox"/>	3	<input checked="" type="checkbox"/>

At the bottom of the interface, the "MCUs Selection" tab is active, showing the selected MCU: STM32G0, Series: STM32G0x0 Value line, Mcu: STM32G070RBTx, Package: LQFP64, and Required Peripherals: None.

DMA configuration panel

62

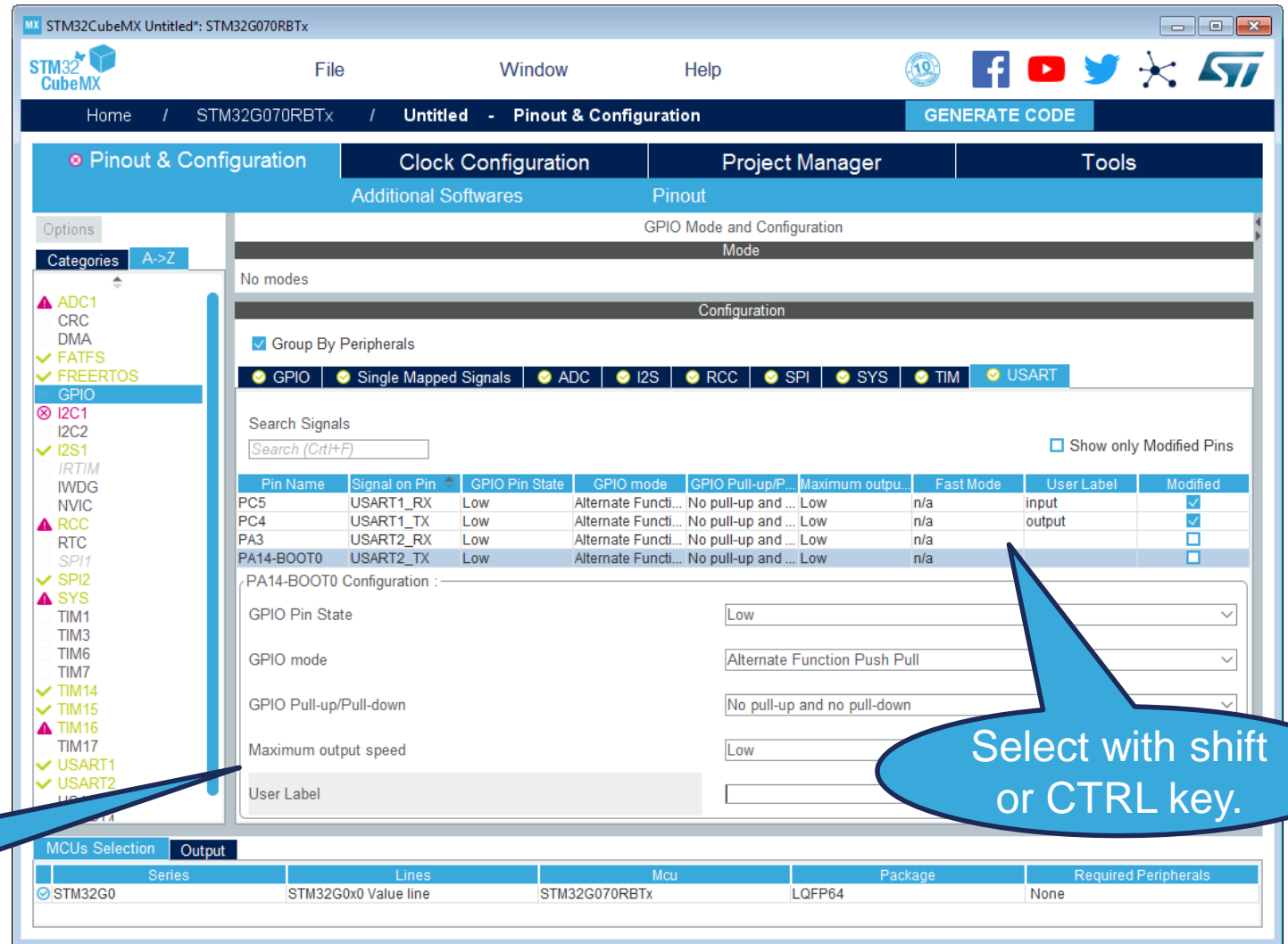
- Manages all DMA requests including memory to memory.
- Configure direction, priority and other settings.

The screenshot displays the STM32CubeMX software interface for configuring the DMA controller. The main window title is 'STM32CubeMX Untitled*: STM32G070RBTx'. The top menu bar includes 'File', 'Window', and 'Help'. The breadcrumb navigation shows 'Home / STM32G070RBTx / Untitled - Pinout & Configuration'. The 'GENERATE CODE' button is visible in the top right. The left sidebar shows a tree view of components, with 'DMA' selected. The main area is divided into 'Options', 'Categories', and 'Configuration' sections. The 'Configuration' section is active, showing 'DMA1' and 'MemToMem' selected. The 'DMA Request Generator Settings' section includes fields for 'Request generation Signal', 'Signal polarity', and 'Request number'. The 'DMA Request Synchronization Settings' section includes checkboxes for 'Enable synchronization' and 'Enable event', and fields for 'Synchronization signal' and 'Request number'. At the bottom, the 'MCUs Selection' table is visible:

Series	Lines	Mcu	Package	Required Peripherals
STM32G0	STM32G0x0 Value line	STM32G070RBTx	LQFP64	None

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.



Make sure the speed is correct.

Select with shift or CTRL key.

- 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.



```
main.c
22  .....
23  */
24  /* Includes -----
25  #include "stm32f4xx_hal.h"
26  #include "cmsis_os.h"
27  #include "lwip.h"
28  #include "usb_device.h"
29
30  /* Define structures */
31  ADC_HandleTypeDef hadc1;
32
33
34  /* USER CODE BEGIN 0 */
35
36  /* USER CODE END 0 */
37  /* Private function prototypes -----
38  static void SystemClock_Config(void);
39  static void StartThread(void const * argument);
40  static void MX_GPIO_Init(void);
41  static void MX_ADC1_Init(void);
42  static void MX_NVIC_Init(void);
43
44  int main(void)
45  {
46  /* USER CODE BEGIN 1 */
47
48  /* USER CODE END 1 */
49  /* MCU Configuration-----
50  /* Reset of all peripherals, Initializes the Flash interfa
51  HAL_Init();
52  /* Configure the system clock */
```

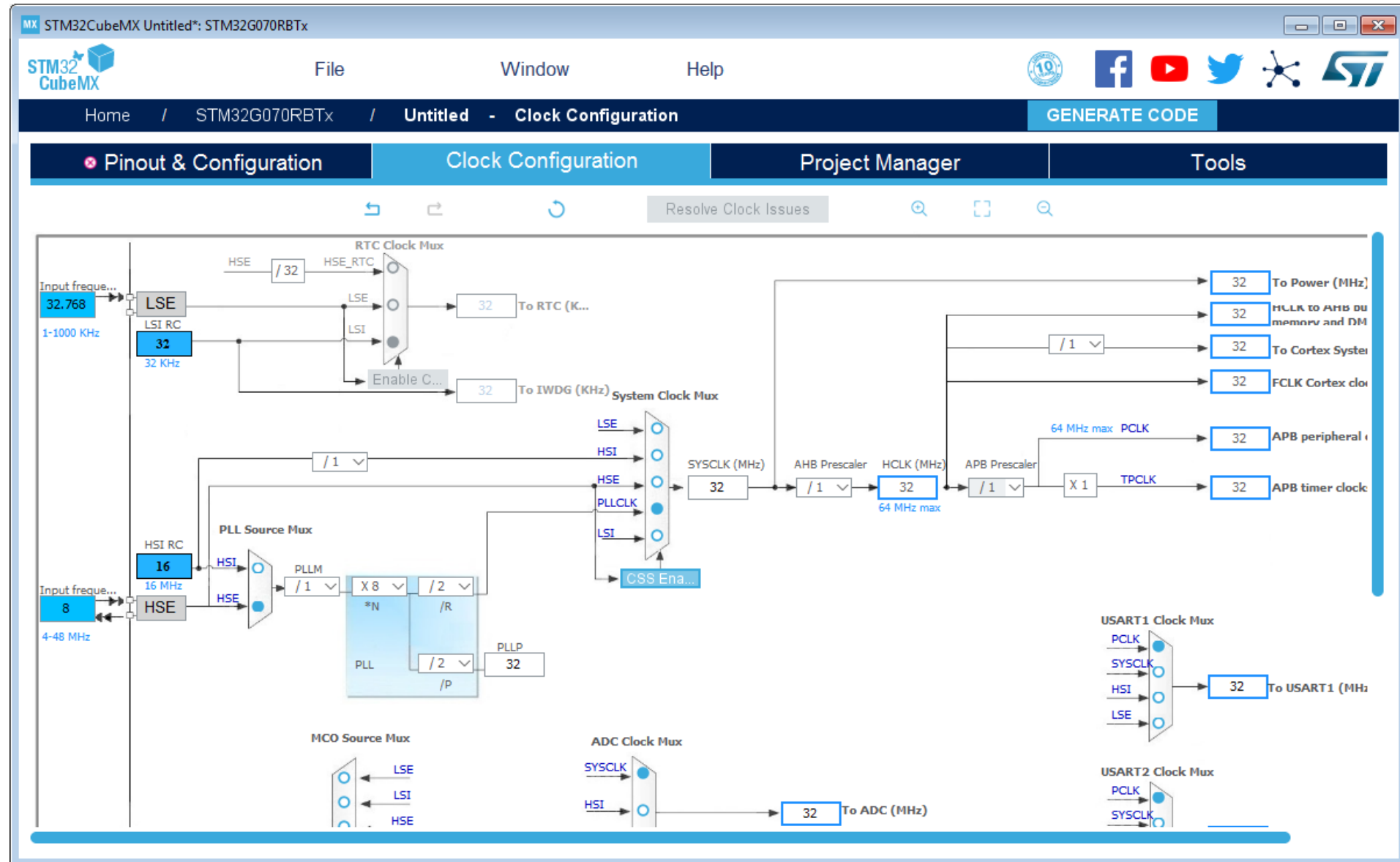
Write your code here to keep option to regenerate the project

Ln:1 Col:1 Sel:0 Dos\Windows ANSI INS

Clock configuration

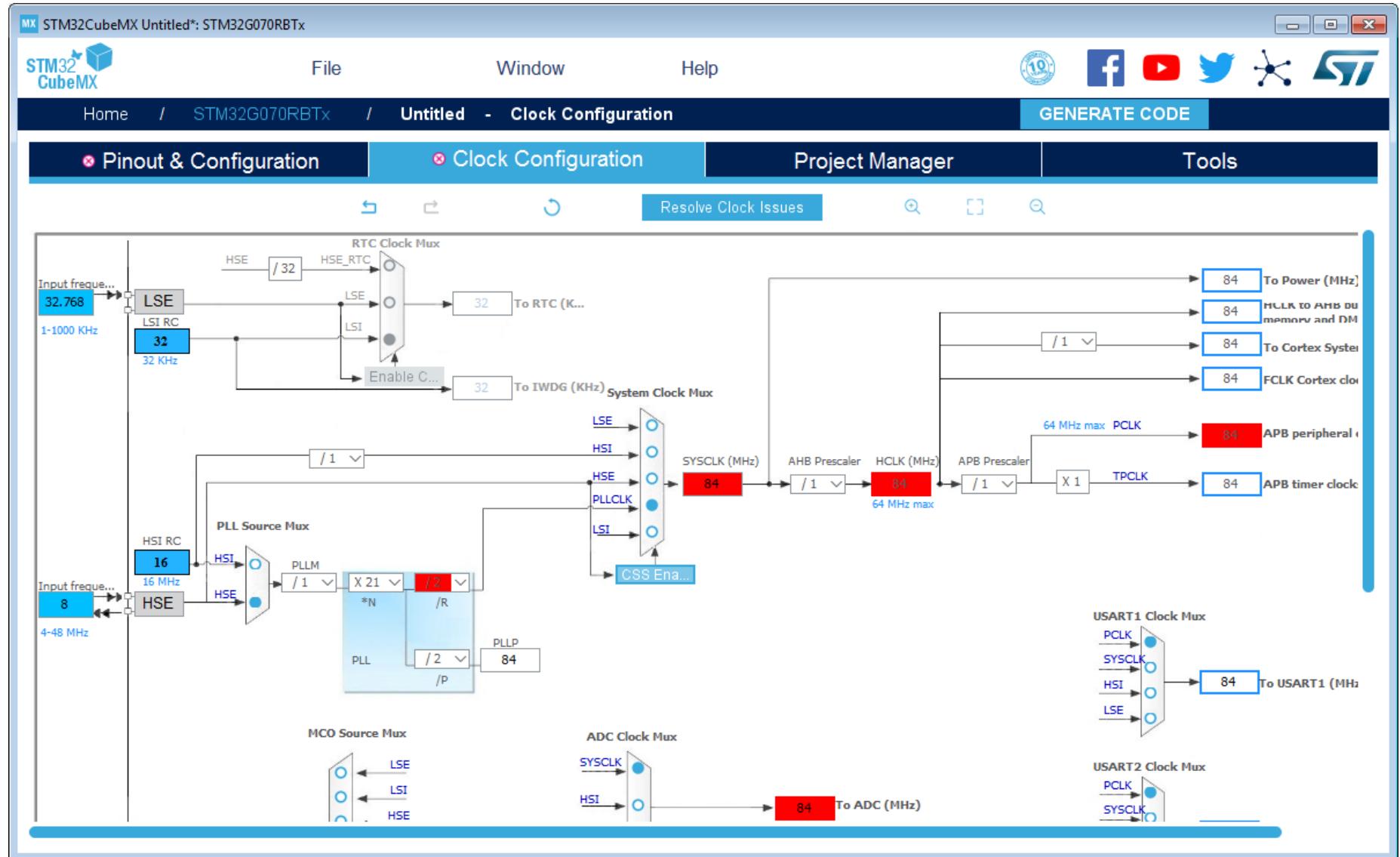
65

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



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.



Code generation project settings

67

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

The screenshot displays the STM32CubeMX Project Manager interface. The window title is "STM32CubeMX Untitled*: STM32G070RBTx". The menu bar includes "File", "Window", and "Help". The breadcrumb navigation shows "Home / STM32G070RBTx / Untitled - Project Manager". A "GENERATE CODE" button is visible in the top right. The main interface has tabs for "Pinout & Configuration", "Clock Configuration", "Project Manager" (selected), and "Tools". A "Generate Report" button is located above the settings panel. The settings panel is divided into sections: "Project" (Project Name: "Great application", Project Location: "C:\Users\...\Documents\hello"), "Code Generator" (Application Structure: "Basic", checkbox "Do not generate the main()"), "Advanced Settings" (Toolchain Folder Location: "C:\Users\...\Documents\hello\Great application\", Toolchain / IDE: "EWARM V8", checkbox "Generate Under Root"), "Linker Settings" (Minimum Heap Size: "0x200", Minimum Stack Size: "0x400"), and "Mcu and Firmware Package" (Mcu Reference: "STM32G070RBTx", Firmware Package Name and Version: "STM32Cube FW_G0 V0.9.0", checkbox "Use latest available version"). At the bottom, there is a table for MCU selection.

MCUs Selection	Output			
Series	Lines	Mcu	Package	Required Peripherals
STM32G0	STM32G0x0 Value line	STM32G070RBTx	LQFP64	None

Code generation options

68

- 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.

The screenshot displays the STM32CubeMX software interface for a project named 'STM32G070RBTx'. The 'Project Manager' tab is active, showing the 'Generate Report' button and the 'Generate Code' button. The 'Code Generator' section is expanded, showing the following options:

- STM32Cube Firmware Library Package:**
 - Copy all used libraries into the project folder
 - Copy only the necessary library files
 - Add necessary library files as reference in the toolchain project configuration file
- Generated files:**
 - Generate peripheral initialization as a pair of '.c/.h' files per peripheral
 - Backup previously generated files when re-generating
 - Keep User Code when re-generating
 - Delete previously generated files when not re-generated
- HAL Settings:**
 - Set all free pins as analog (to optimize the power consumption)
 - Enable Full Assert
- Template Settings:**
 - Select a template to generate customized code
 - Settings...

At the bottom, the 'MCUs Selection' tab is active, showing a table with the following data:

Series	Lines	Mcu	Package	Required Peripherals
<input checked="" type="checkbox"/> STM32G0	STM32G0x0 Value line	STM32G070RBTx	LQFP64	None

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

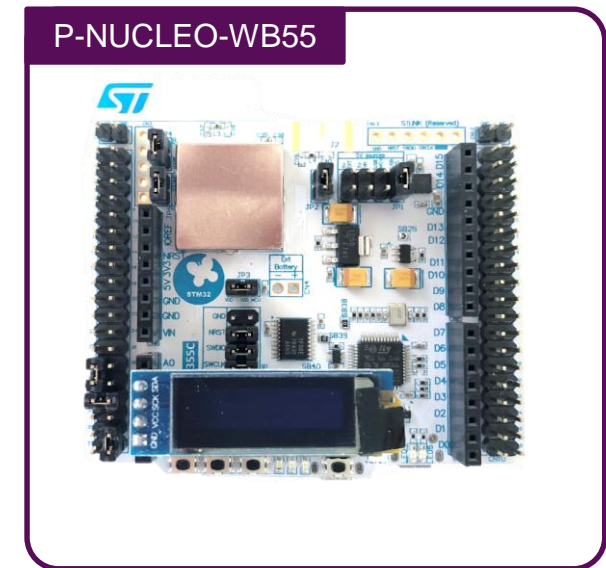
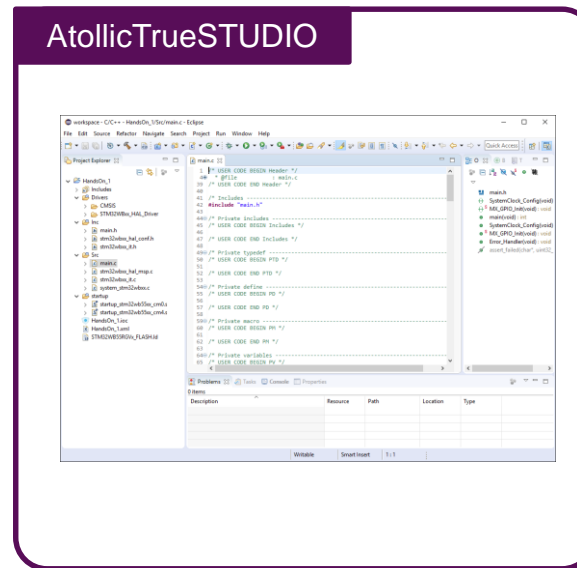
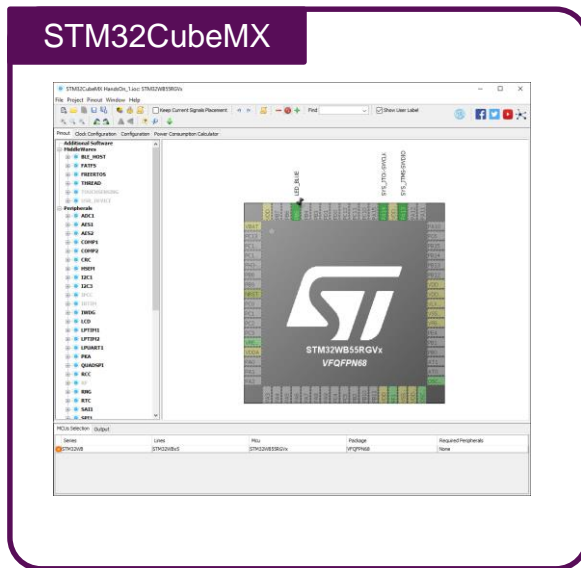
Let's blink an LED



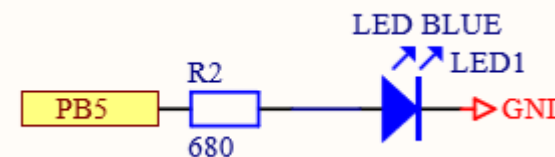


Hands-On principle

- 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



Toggle @1Hz rate

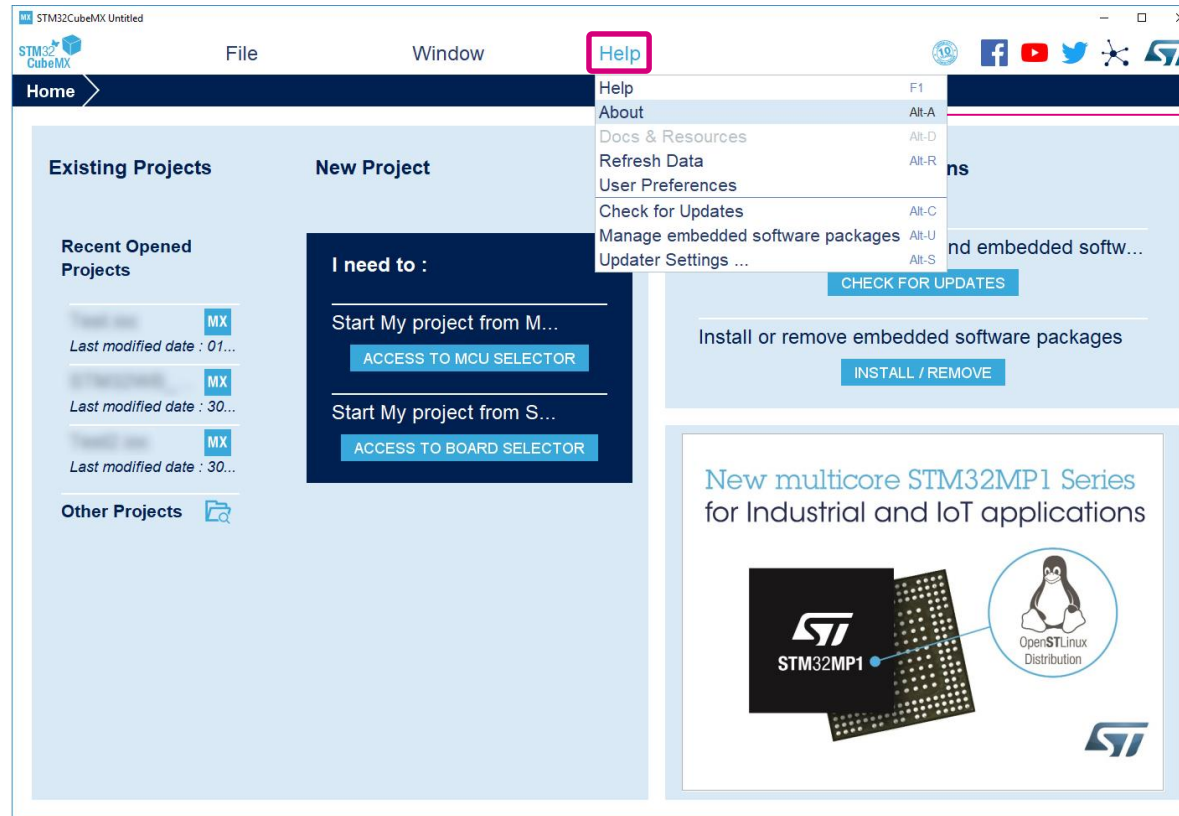




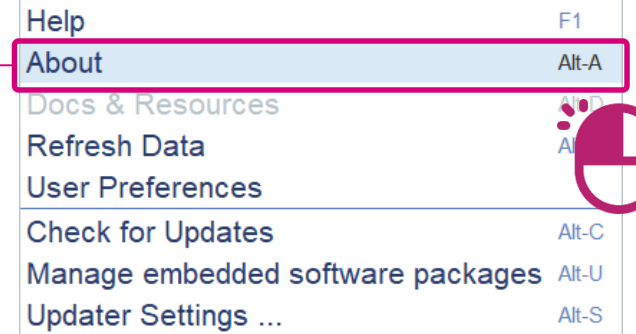
Open STM32CubeMX

1

MX STM32CubeMX
Desktop app



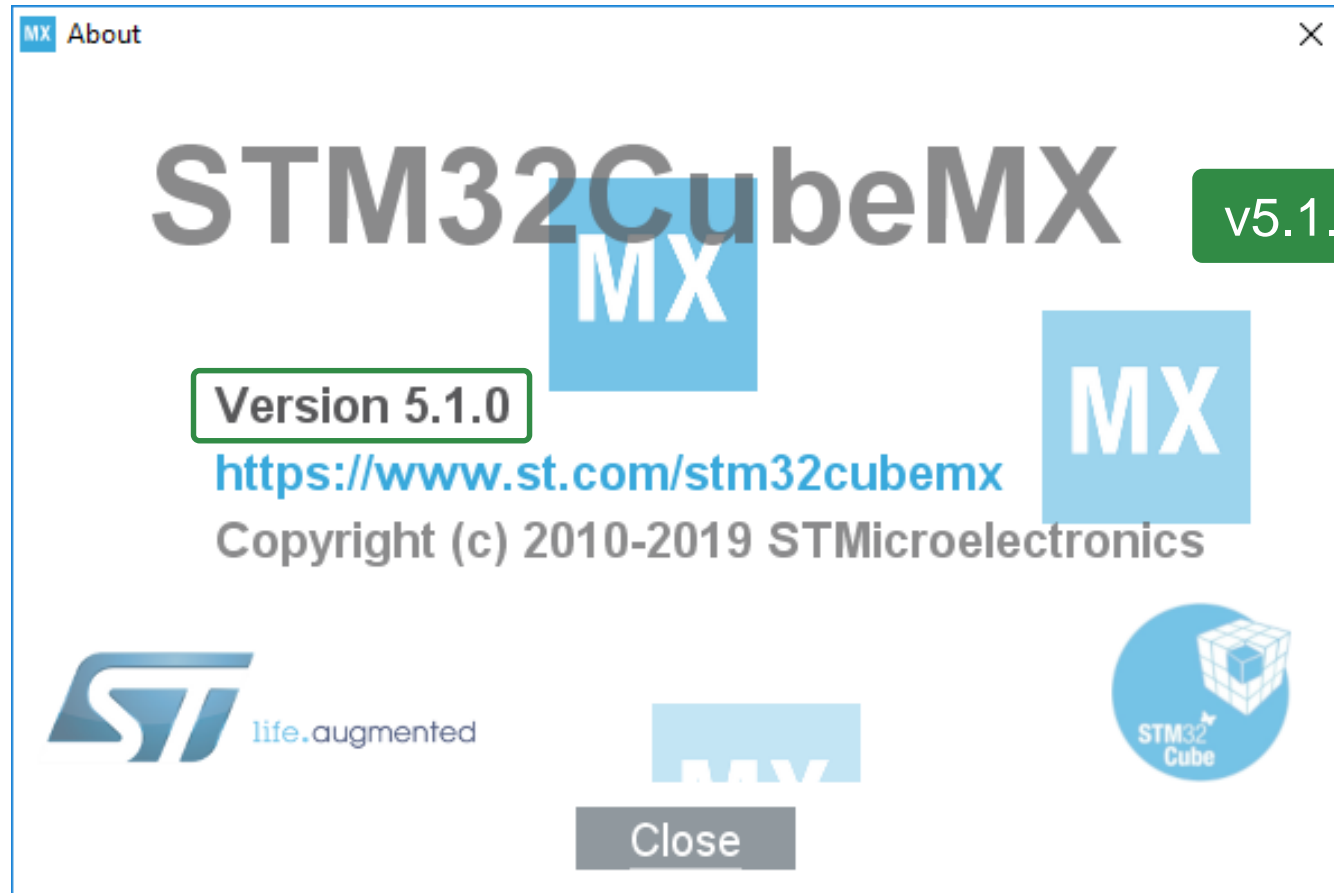
Help



Help → About
[Alt+A]



Check the STM32CubeMX version



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



2

File

- New Project ... Ctrl-N
- Load Project ... Ctrl-L
- Import Project ... Ctrl-I
- Save Project Ctrl-S
- Save Project As .. Ctrl-A
- Close Project Ctrl-C
- Generate Report Ctrl-R
- Recent Projects
- Exit Ctrl-X

Start My project from M...
ACCESS TO MCU SELECTOR

Start My project from S...
ACCESS TO BOARD SELECTOR

Manage software installations

Check for STM32CubeMX and embedded softw...
CHECK FOR UPDATES

Install or remove embedded software packages
INSTALL / REMOVE

New multicore STM32MP1 Series
for Industrial and IoT applications

STM32MP1

OpenSTLinux
Distribution

File → New Project...

[Ctrl+N]

(or click on ACCESS TO MCU SELECTOR)



1

STM32WB55RG

Select the sales type

MX New Project

MCU Selector Board Selector

MCU Filters

Part Number Search

STM32WB55RG

Core

Series

Line

Package

Other

Price = 0.0

IO = 49

Eeprom = 0 (Bytes)

Features Block Diagram Docs & Resources Datasheet Buy Start Project

STM32WB55RG

STM32WB

VFQFPN68

STM32WB55RGVx

MCUs List: 1 item

+ Display similar items

*	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

2



*	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



Let's start with Pinout configuration

STM32CubeMX Untitled: STM32WB55RGVx

File Window Help

Home > STM32WB55RGVx > Untitled - Pinout & Configuration > GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

Additional Softwares Pinout

Options [Search] Categories A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware >

Pinout view System view

STM32WB55RGVx
VFQFPN68

MCUs Selection Output

Series	Lines	Mcu	Package	Required Peripherals
STM32WB	STM32WBx5	STM32WB55RGVx	VFQFPN68	None



Configure Debug interface

1 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 >

SYS Mode and Configuration

Mode

Debug **Serial Wire**

- System Wake-Up 1
- System Wake-Up 2
- System Wake-Up 3
- System Wake-Up 4
- System Wake-Up 5

Power Voltage Detector In

VREFBUF Mode

Timebase Source

Configuration

Warning: This IP has no parameters to be configured

2

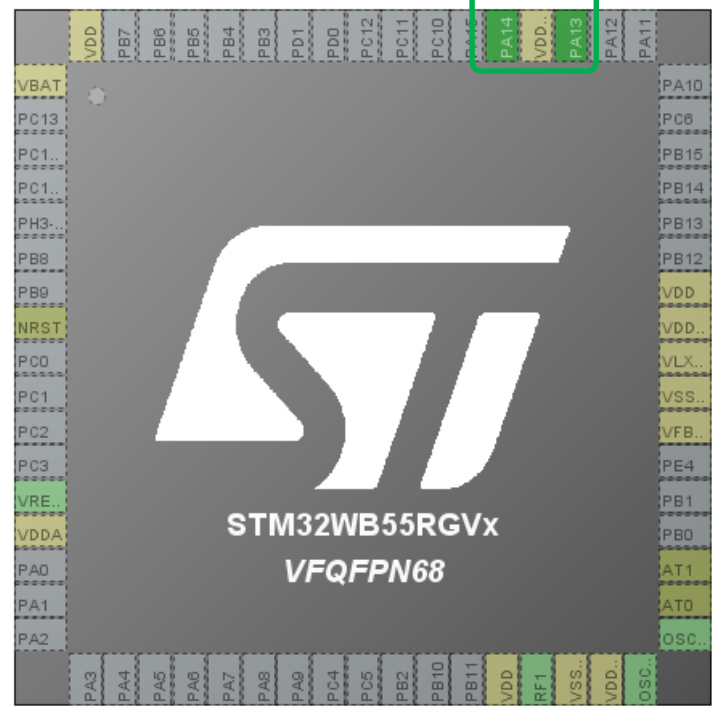
3

Pinout view System view



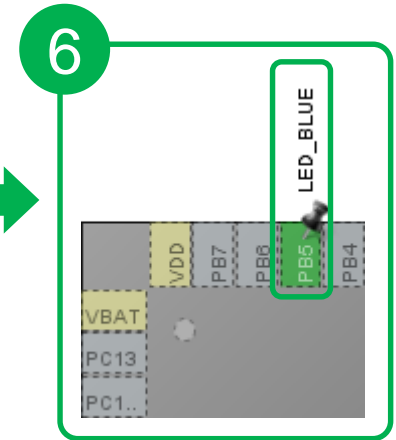
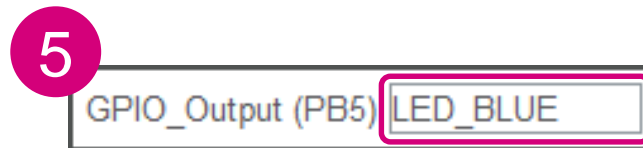
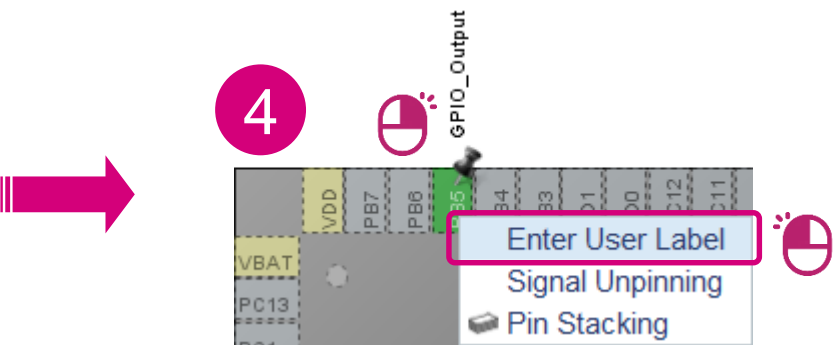
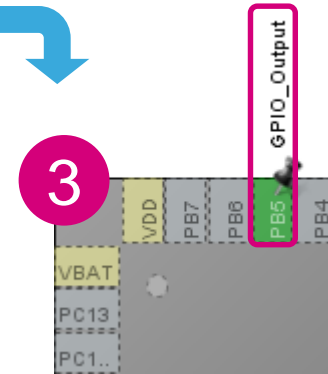
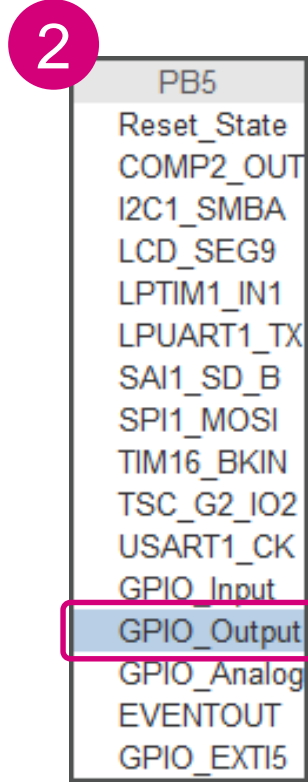
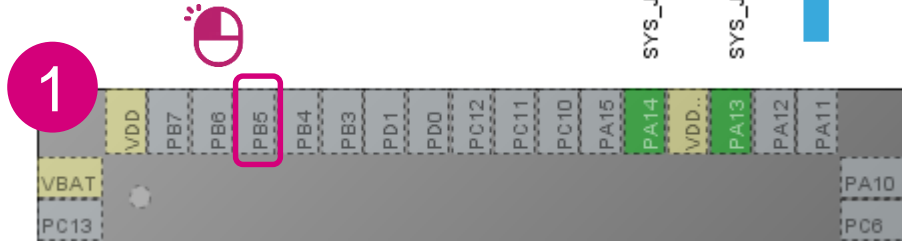
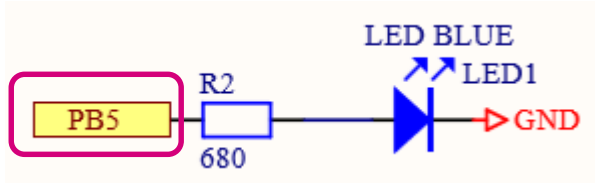
PA14 SYS_JTCK-SWCLK

PA13 SYS_JTMS-SWDIO





Configure the pin for blue LED control

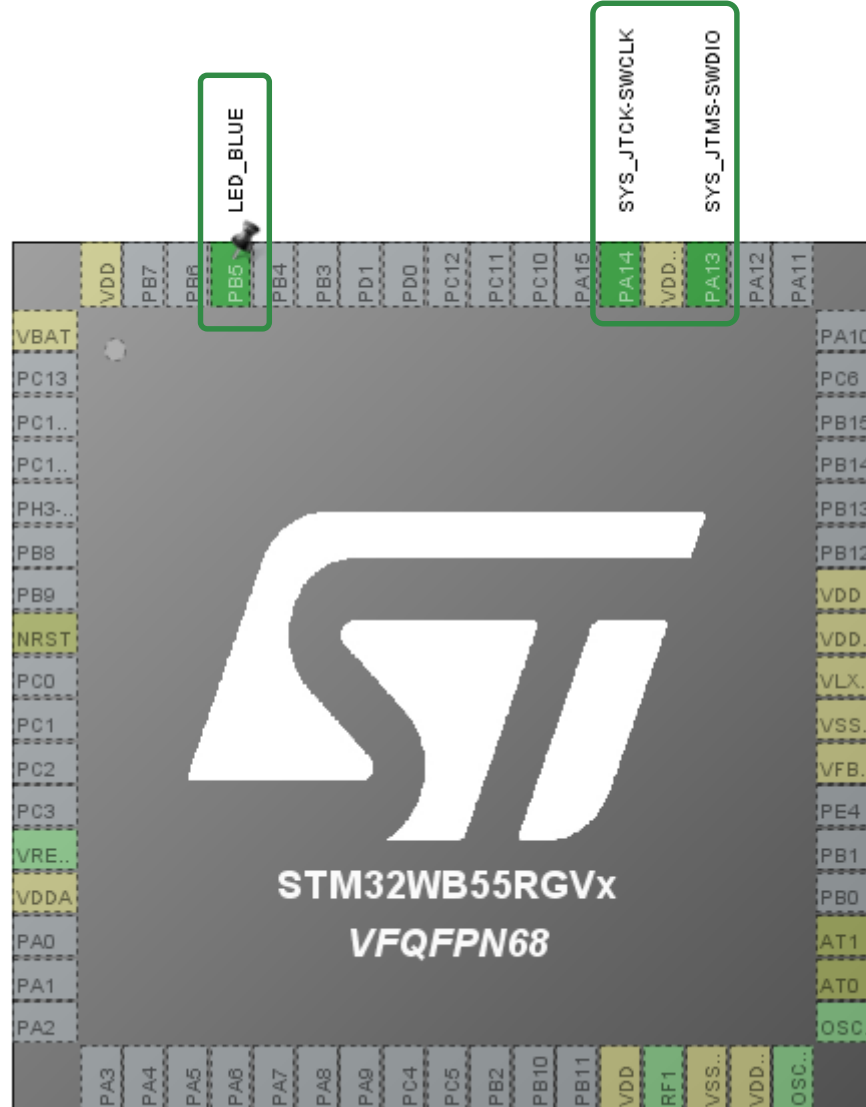




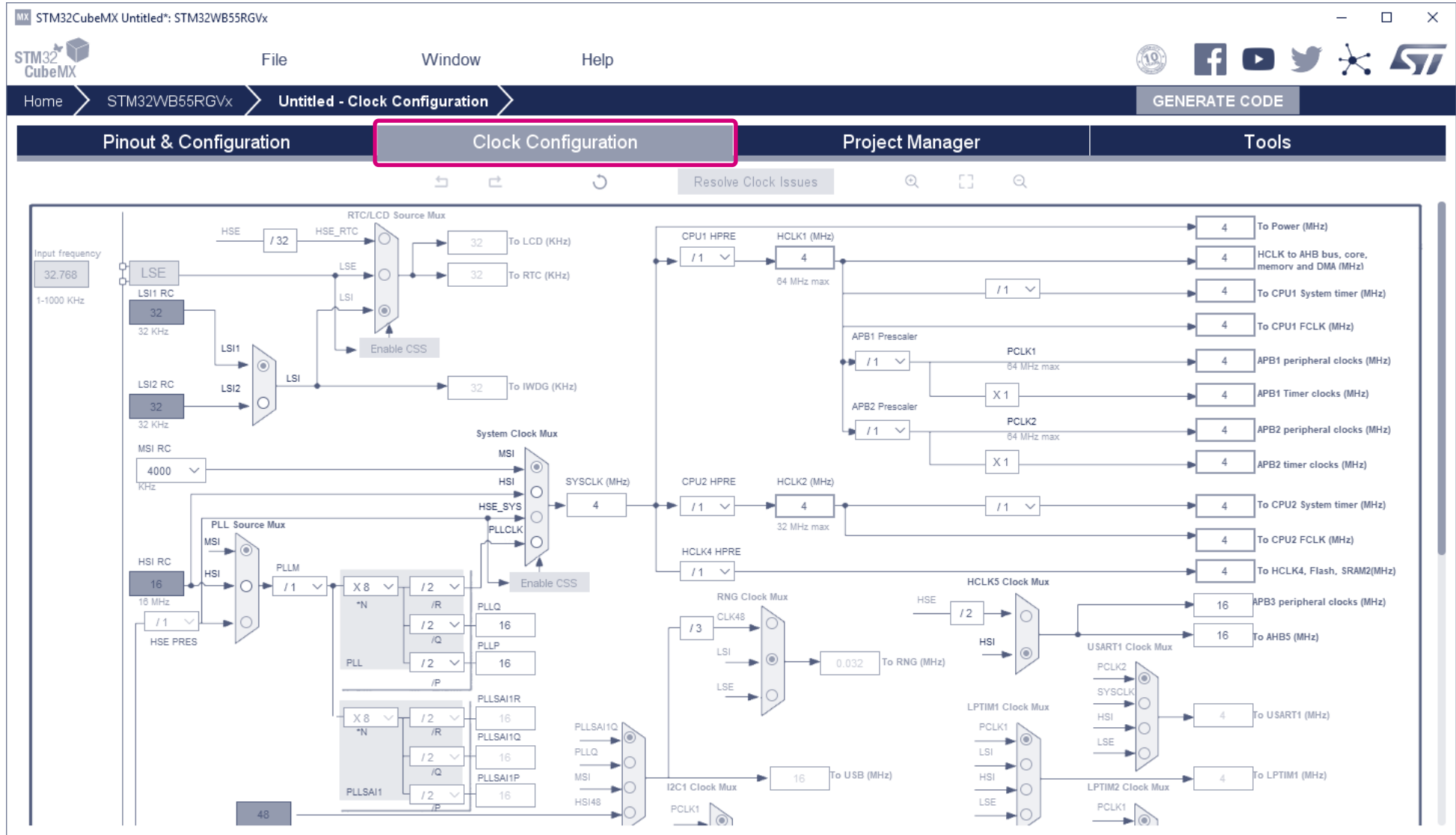
Checkpoint



SWCLK @PA14
SWDIO @PA13
LED_BLUE @PB5



Now switch to clock configuration

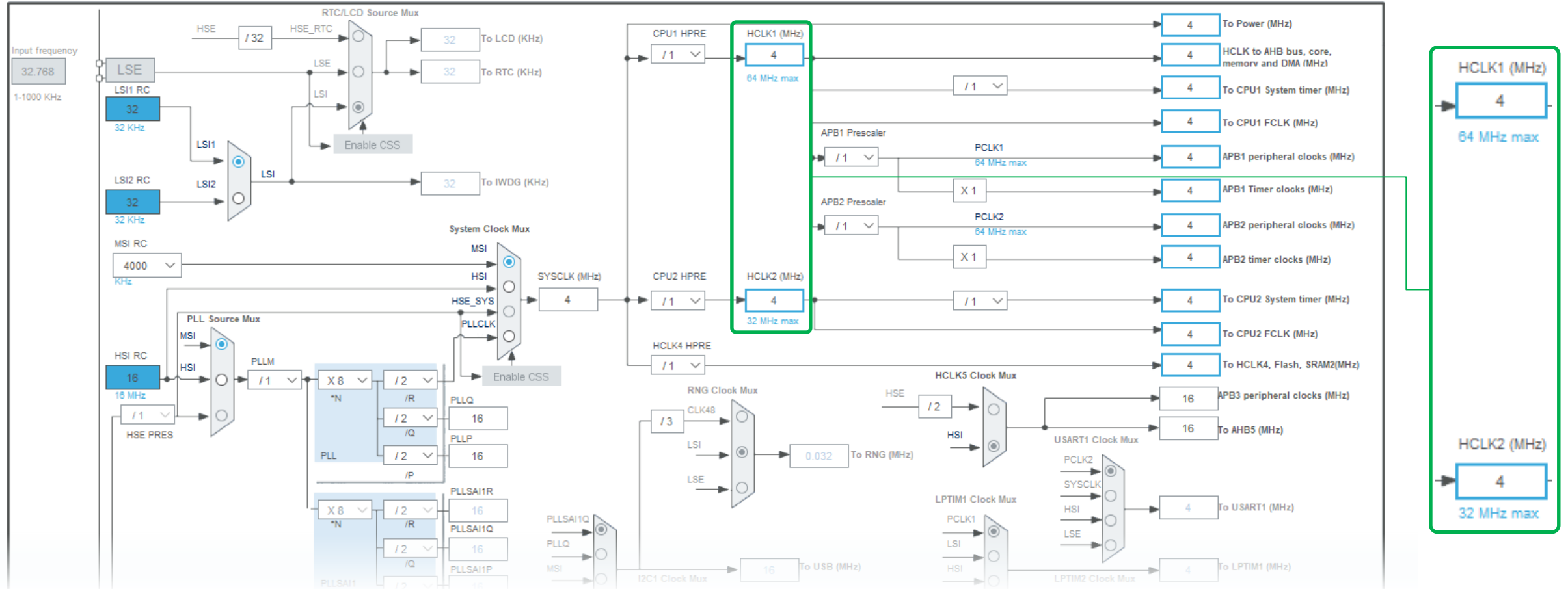




1

Clock Configuration

Clock configuration



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





Finalize the project settings

The screenshot shows the STM32CubeMX Project Manager window. The 'Project Manager' tab is selected and highlighted with a pink border. The window is divided into four main sections: Pinout & Configuration, Clock Configuration, Project Manager, and Tools. The Project Manager section is further divided into Project, Code Generator, and Advanced Settings.

Project Settings:

- Project Name: HandsOn_1
- Project Location: C:\STM32WB_workshop
- Application Structure: Basic
- Toolchain Folder Location: C:\STM32WB_workshop\HandsOn_1\
- Toolchain / IDE: TrueSTUDIO

Linker Settings:

- Minimum Heap Size: 0x200
- Minimum Stack Size: 0x400

Mcu and Firmware Package:

- Mcu Reference: STM32WB55RGVx
- Firmware Package Name and Version: STM32Cube FW_WB V1.0.0
- Use Default Firmware Location:
- Firmware Location: C:/Users/stepanew/STM32Cube/Repository/STM32Cube_FW_WB_V1.0.0

MCUs Selection Table:

MCUs Selection	Output	Series	Lines	Mcu	Package	Required Peripherals
<input checked="" type="radio"/>		STM32WB	STM32WBx5	STM32WB55RGVx	VFQFPN68	None



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

Project Manager

The screenshot shows the Project Manager dialog box with the following settings:

- Project:** Project Name: HandsOn_1; Project Location: C:\STM32WB_workshop
- Code Generator:** Application Structure: Basic; Toolchain Folder Location: C:\STM32WB_workshop\HandsOn_1; Toolchain / IDE: TrueSTUDIO
- Advanced Settings:** Linker Settings: Minimum Heap Size: 0x200; Minimum Stack Size: 0x400; Mcu Reference: STM32WB55RGVx; Firmware Package Name and Version: STM32Cube FW_WB V1.0.0

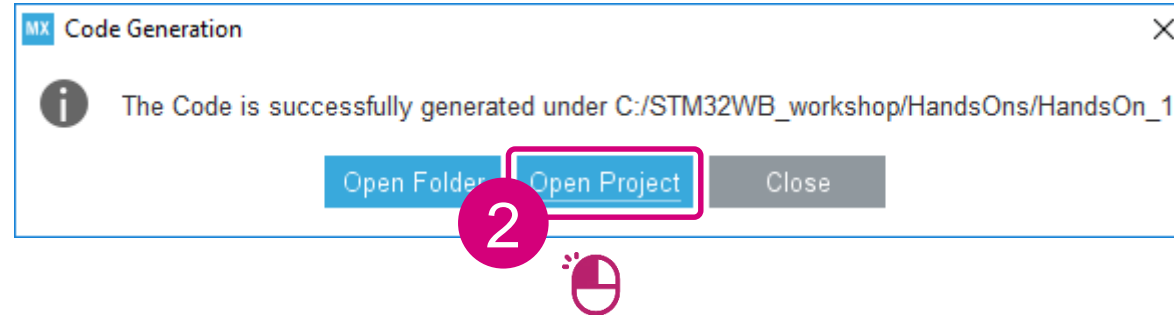
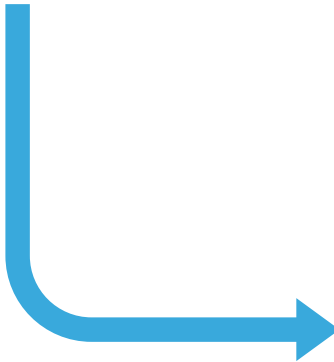
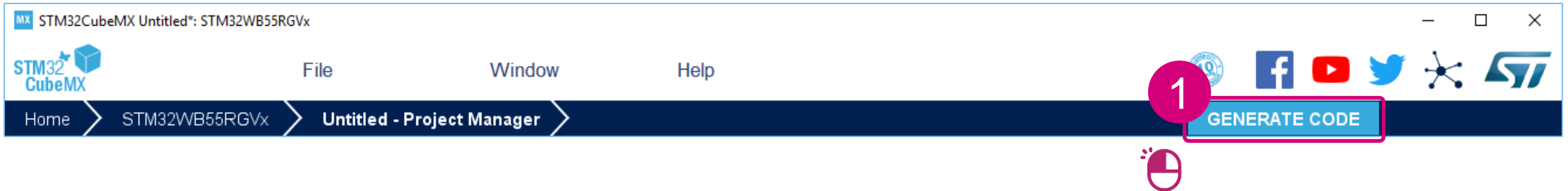
Callouts and annotations:

- 1: Project Settings (tab)
- 2: Project Location (C:\STM32WB_workshop\HandsOns)
- 3: Toolchain / IDE (TrueSTUDIO)
- 4: Firmware Package Name and Version (STM32Cube_FW_WB_V1.0.0)
- Keep all the other options in default state!





Generate the code



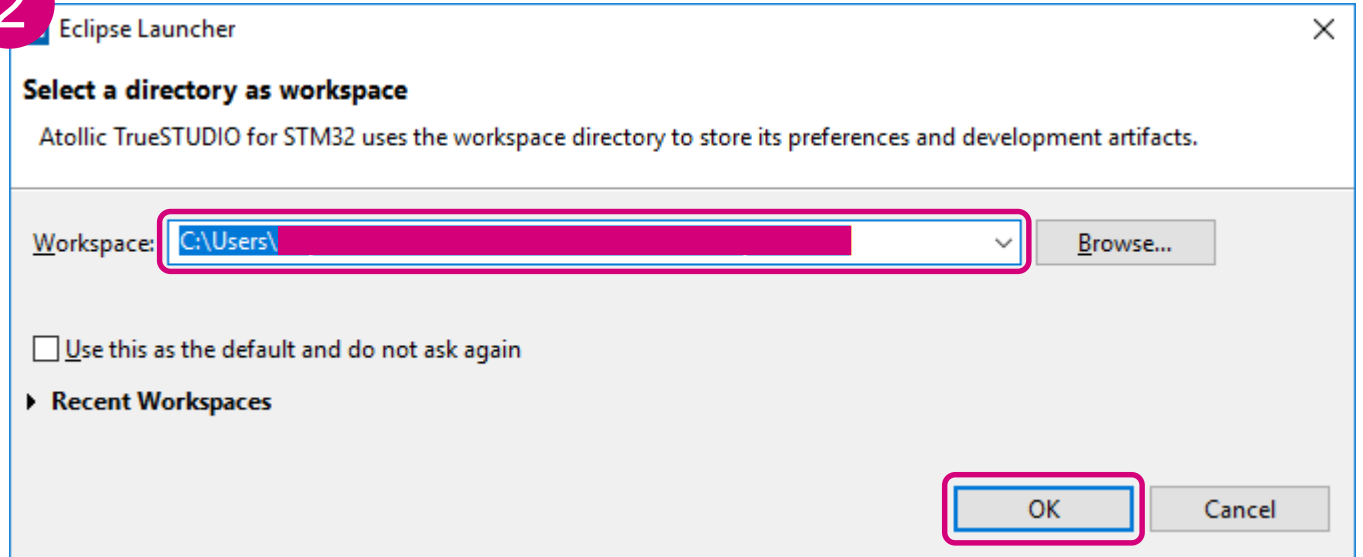
Atollic TrueStudio project opening

1 a atollic



TrueSTUDIO®

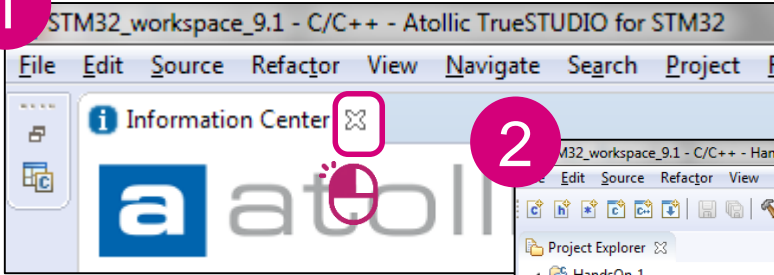
2



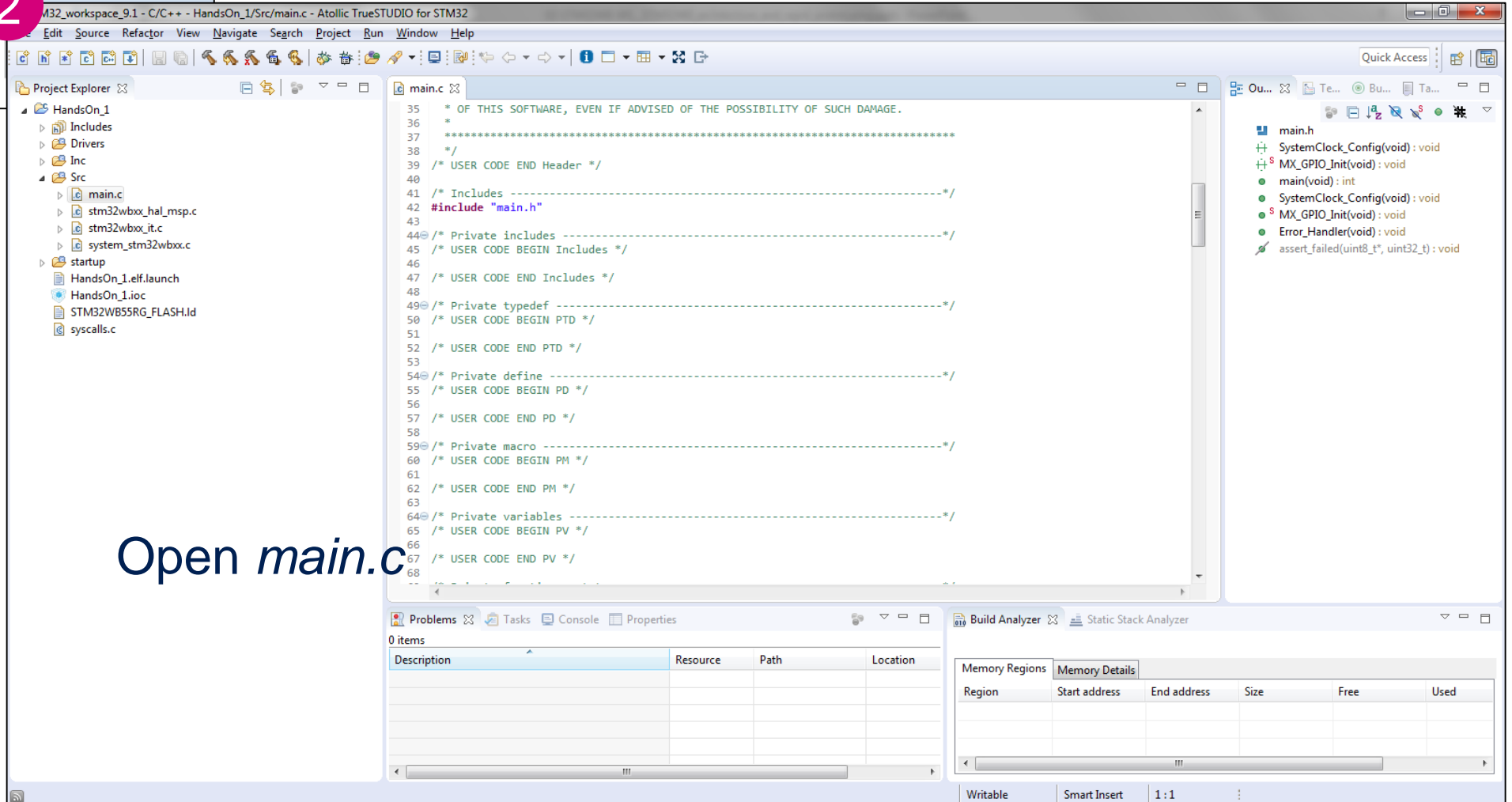


Open the C/C++ perspective

1



2

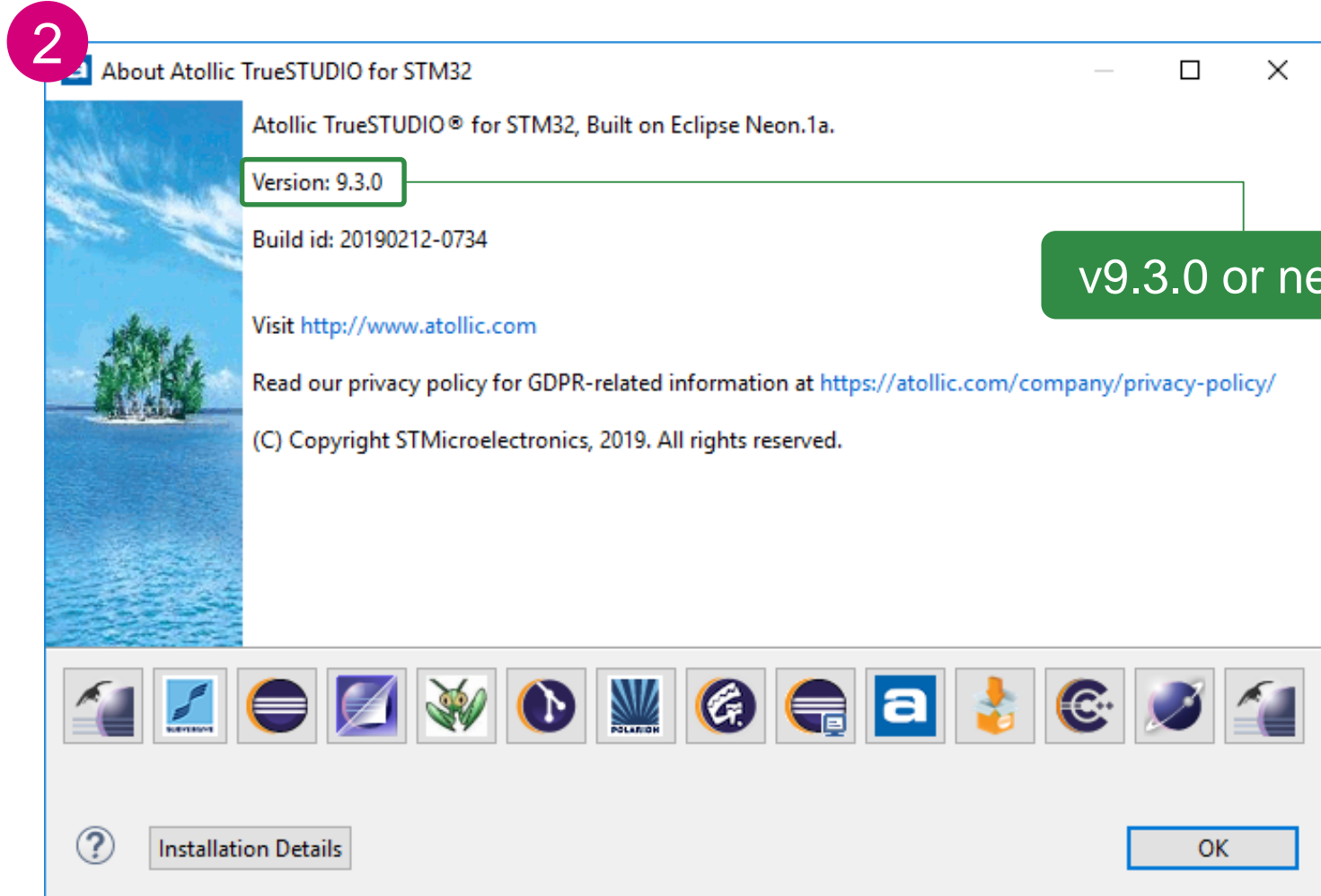
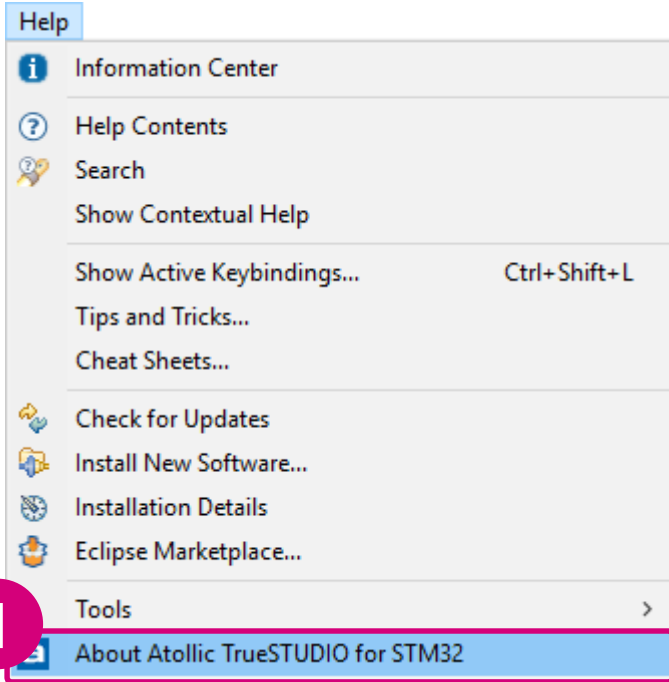


Open *main.c*





Check the Atollic TrueSTUDIO version





Check the Atollic TrueSTUDIO version

TrueSTUDIO NRND Print Save to MyST Share via Email

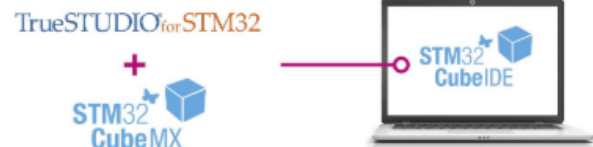
A powerful eclipse based C/C++ integrated development tool for your STM32 projects

[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.

STM32CubeIDE All-in-one STM32 development tool



-> This product is now replaced by STM32CubeIDE

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



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](#)

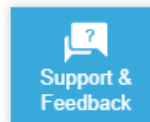
STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem.

STM32CubeIDE 1.0
All-in-one STM32 development tool

The diagram illustrates the integration of TrueSTUDIO for STM32 and STM32CubeMX into the STM32CubeIDE development environment. It shows the logos for both tools on the left, with a plus sign between them, and a red line pointing to a laptop screen on the right that displays the STM32CubeIDE logo.

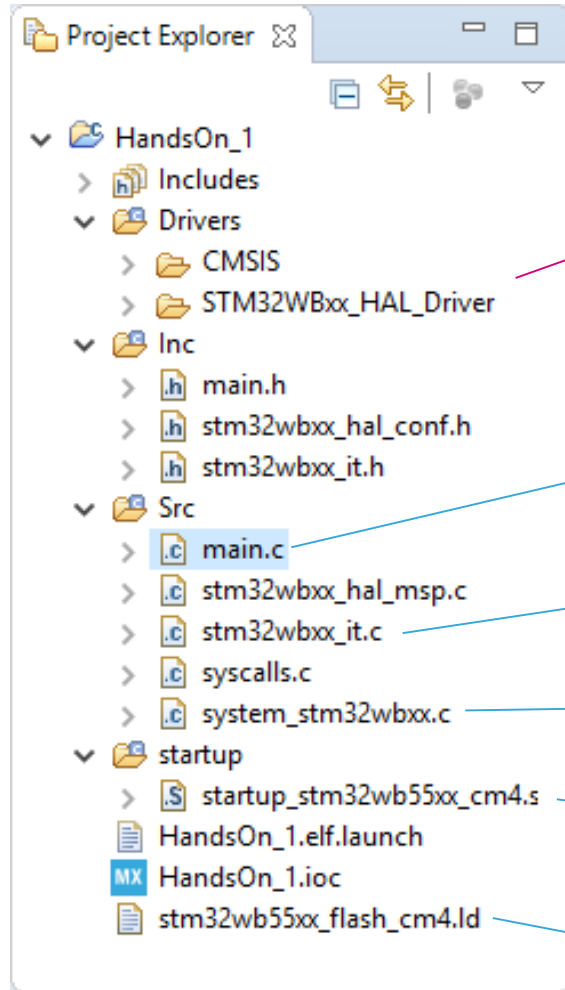
STM32CubeIDE is an advanced C/C++ development platform with IP configuration, code generation, code compilation, and debug features for STM32 microcontrollers. It is based on the ECLIPSE™/CDT framework and GCC toolchain for the development, and GDB for the debugging. It allows the integration of the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





Check out the project tree



ARM CMSIS library
STM32WBxx HAL drivers

Main
Interrupt handlers
Pre-main system configuration
Startup file
Linker file



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





Download the app to the device



1

Connect nucleo board to PC

2



Build
(CTRL+B)

3



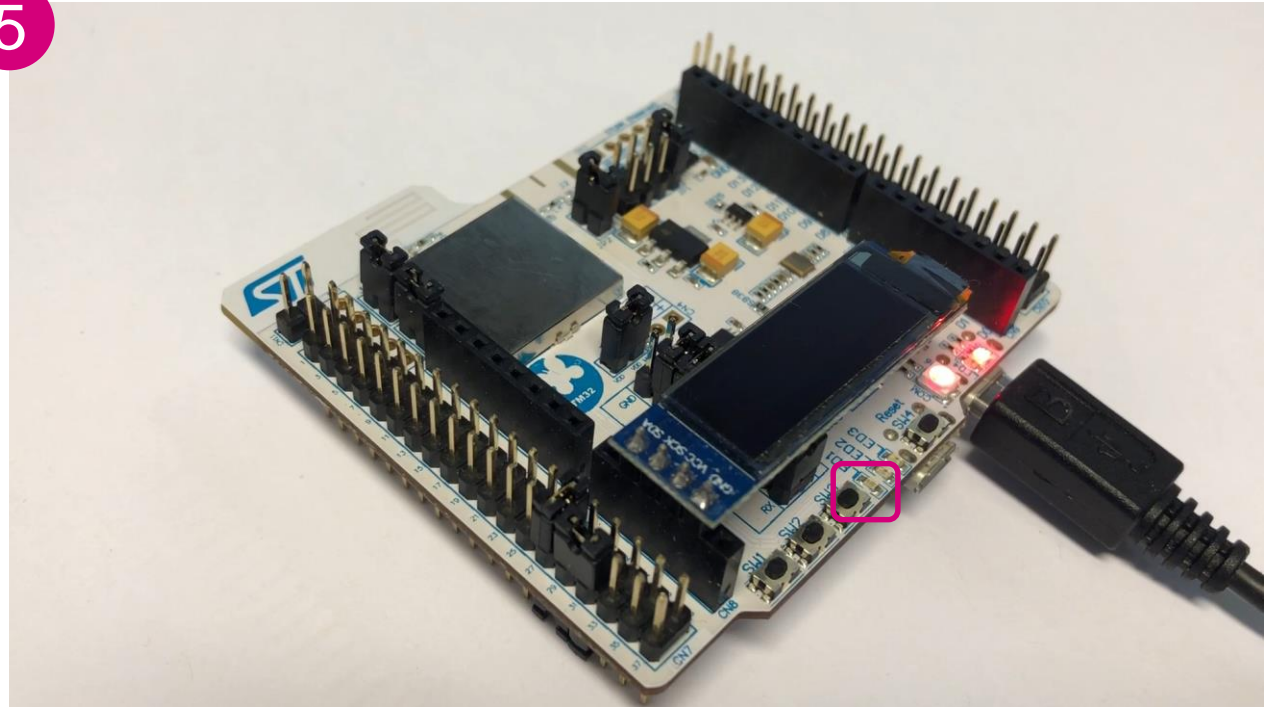
Debug
(F11)

4



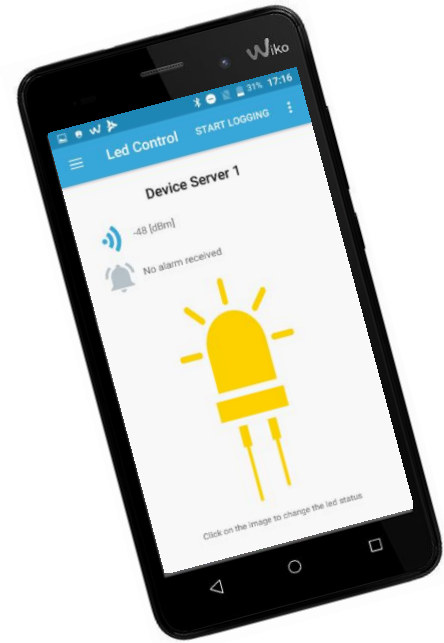
Resume
(F8)

5





Hands-On #2



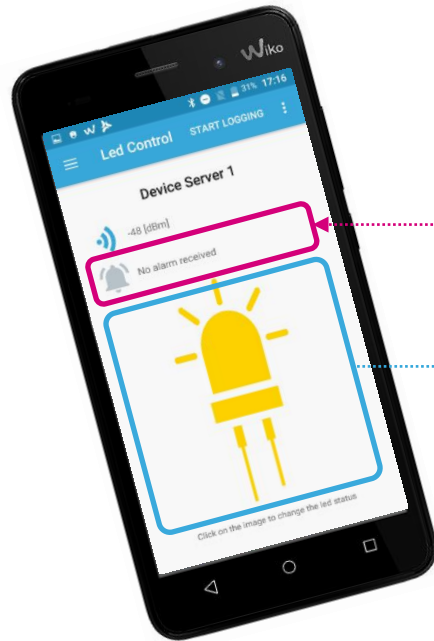


Target

95

Central

GATT Client



Peripheral

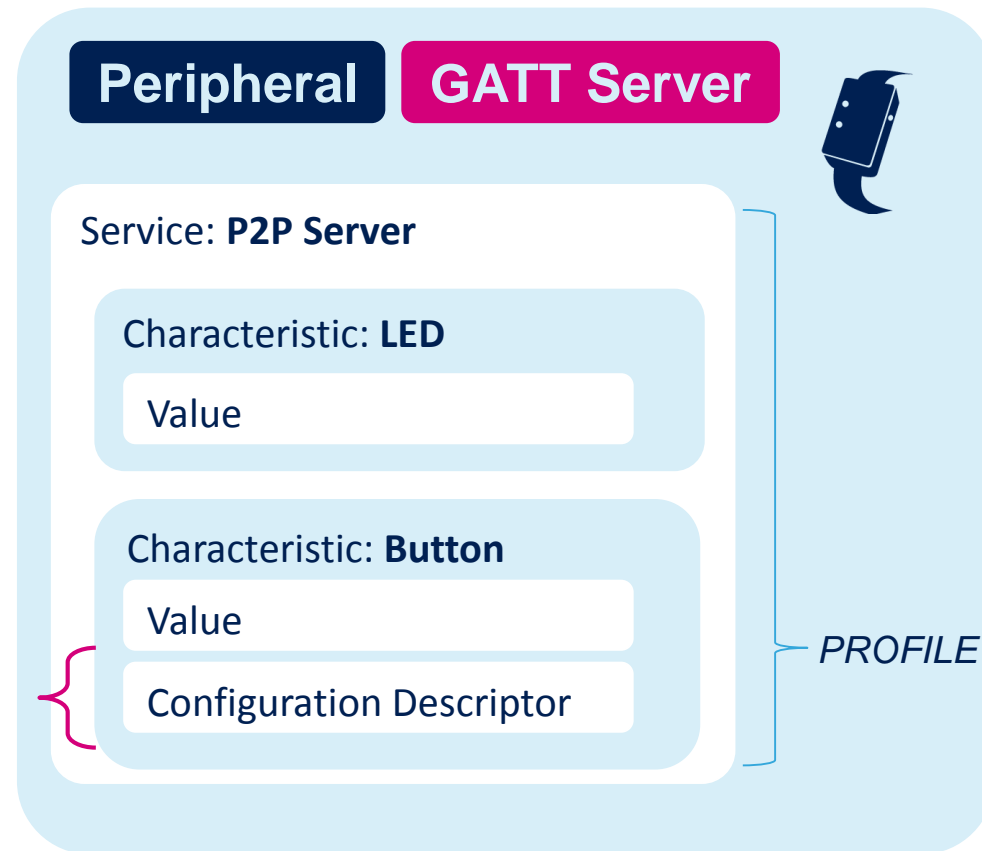
GATT Server





Bluetooth LE basic principles

GATT – Generic Attribute Profile



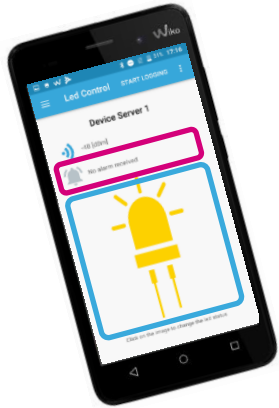
Used in this case by client to enable notification on Button characteristic value change



How should it work?

Central

GATT Client



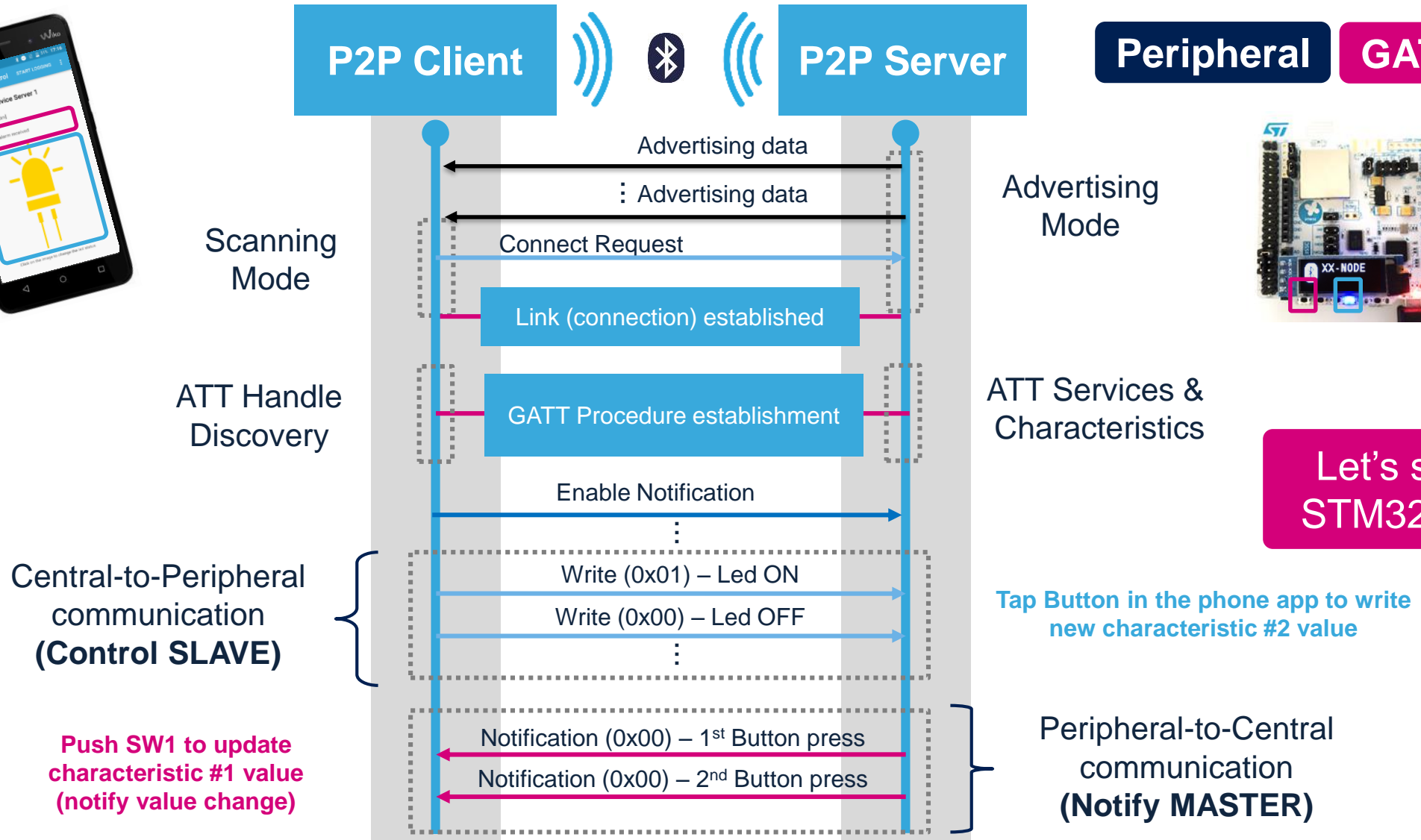
P2P Client



P2P Server

Peripheral

GATT Server



Let's start with STM32CubeMX

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

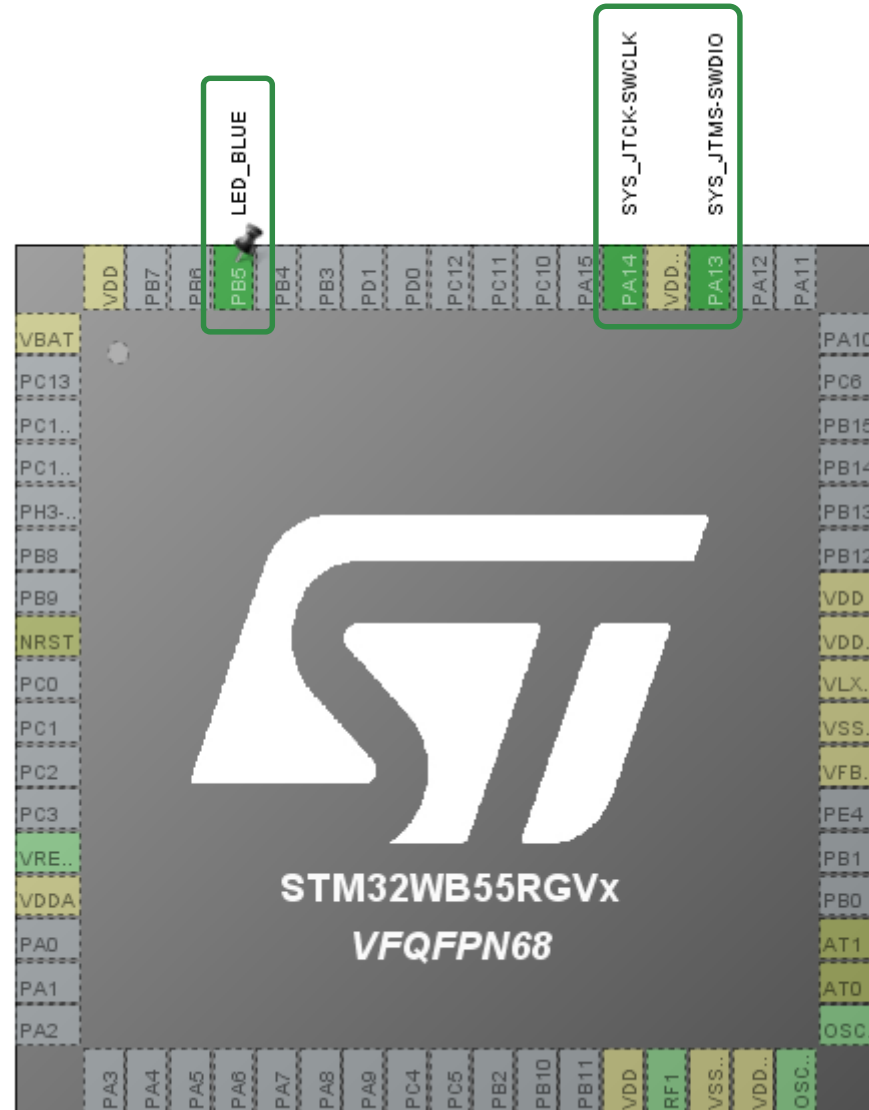
Push SW1 to update characteristic #1 value (notify value change)



Starting point



SWCLK @PA14
SWDIO @PA13
LED_BLUE @PB5

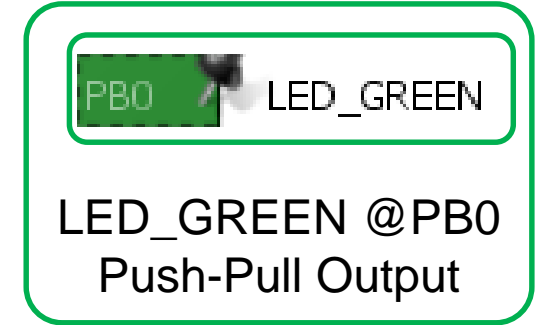
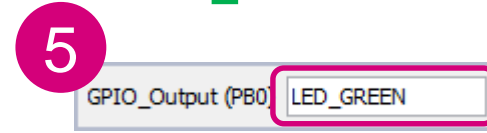
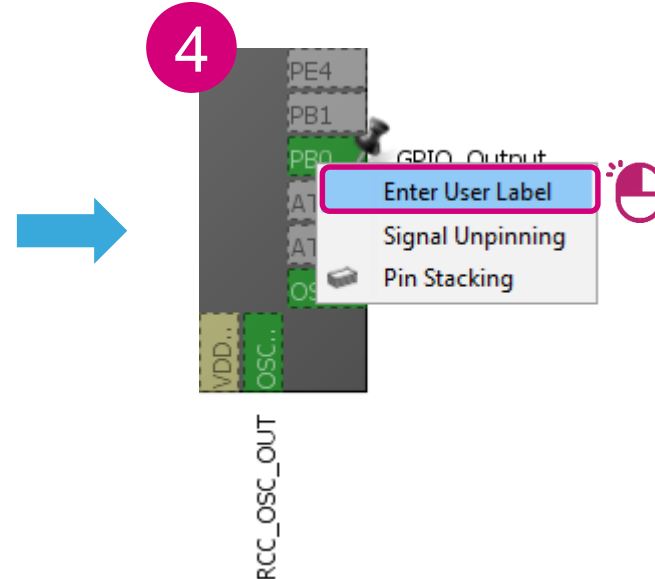
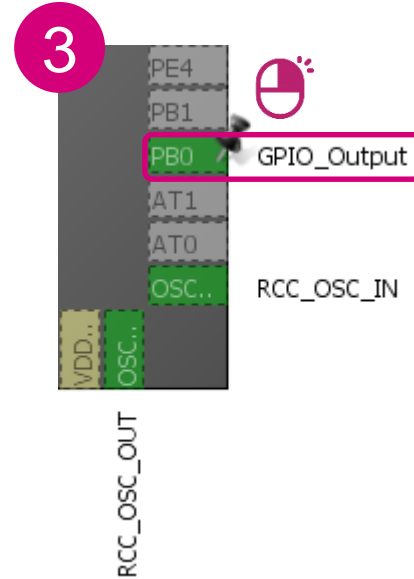
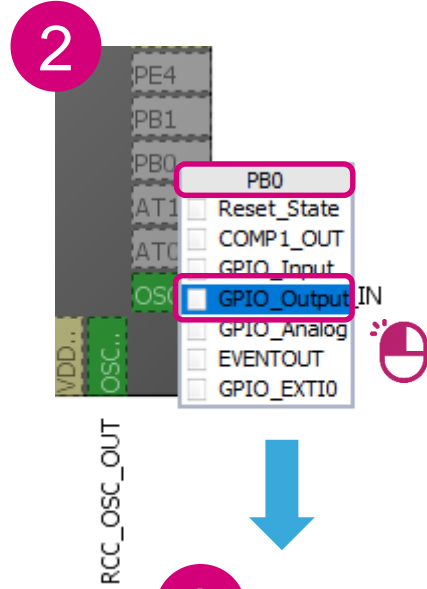
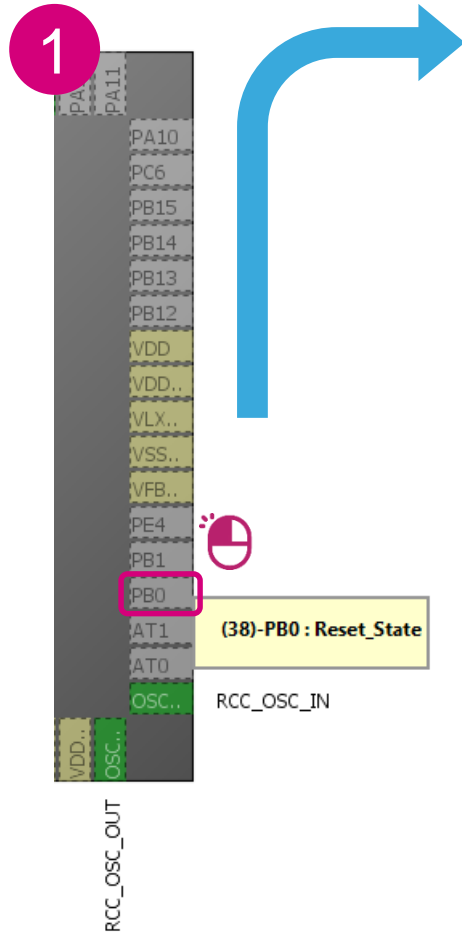
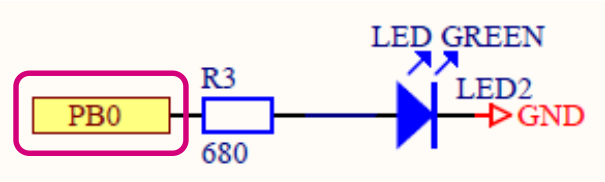


HandsOn_1

Let's do a small time shift!

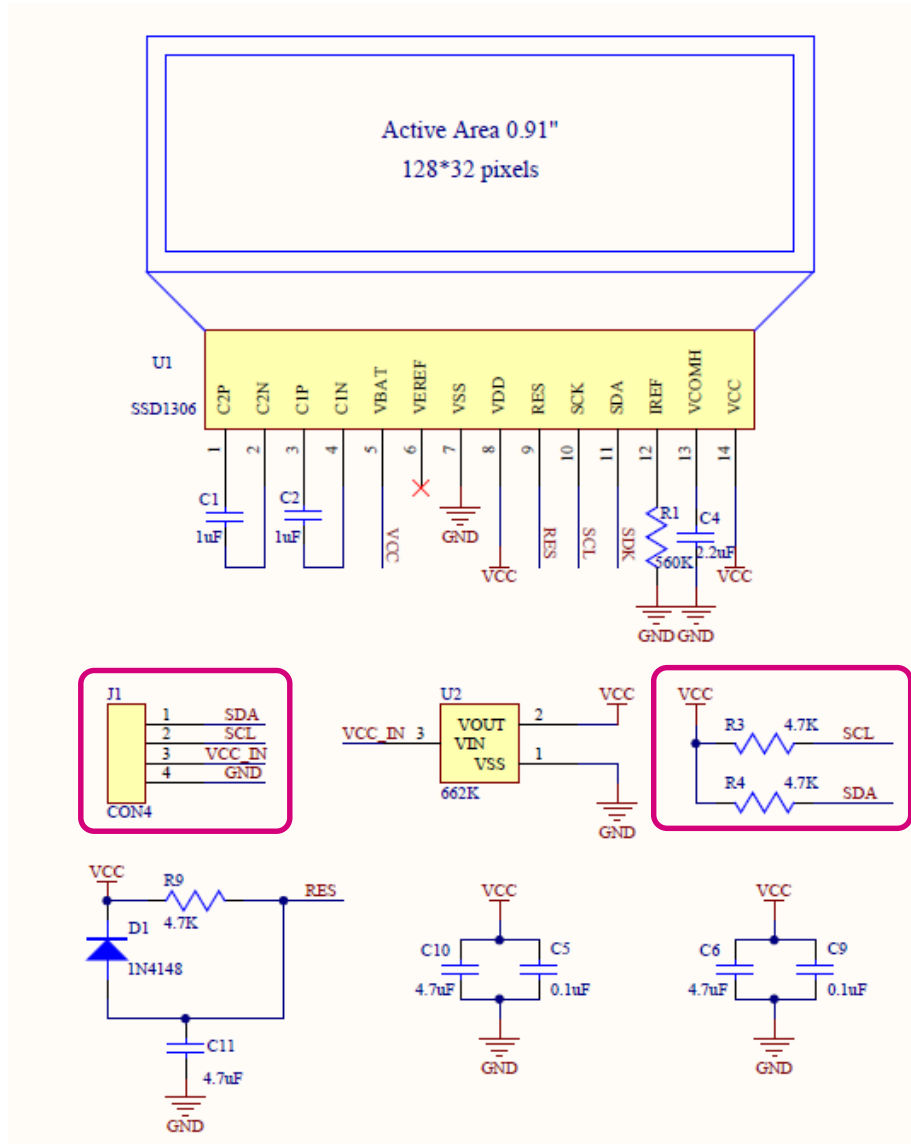
Move to slide 10!!!

Add pin for green LED control



Just for development and debugging purposes.

OLED display connection

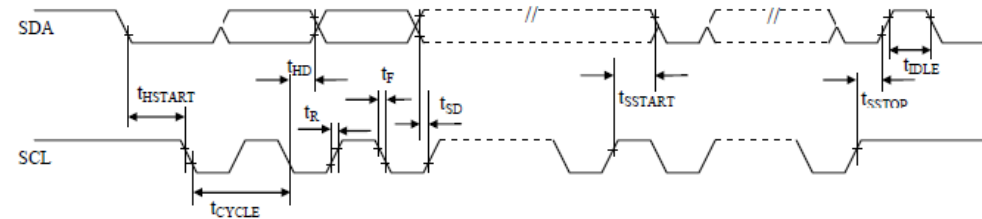


SSD1306 datasheet

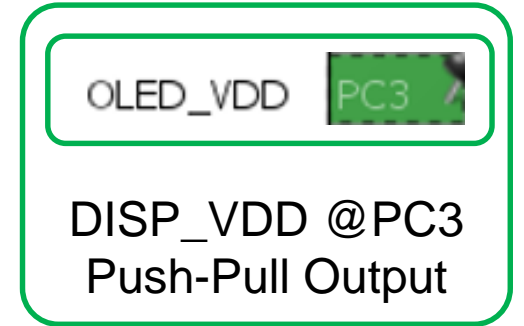
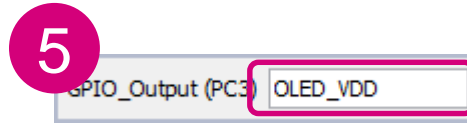
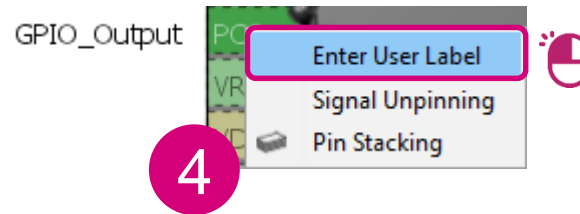
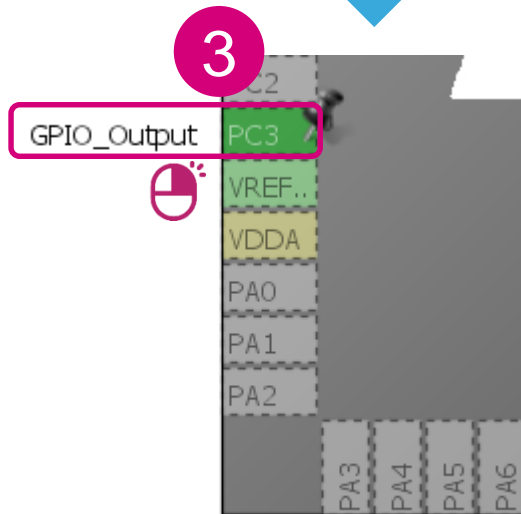
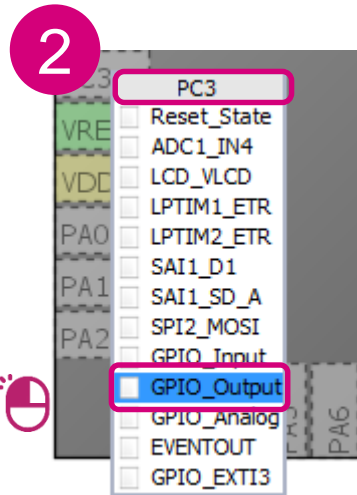
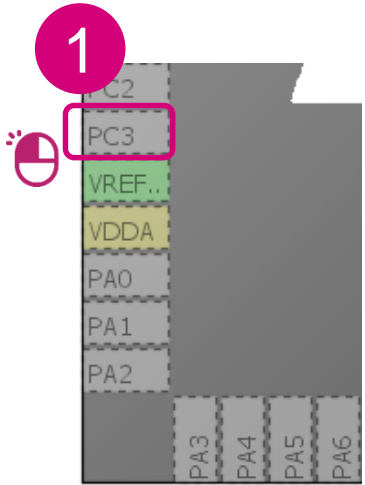
Table 13-6 :I²C Interface Timing Characteristics

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

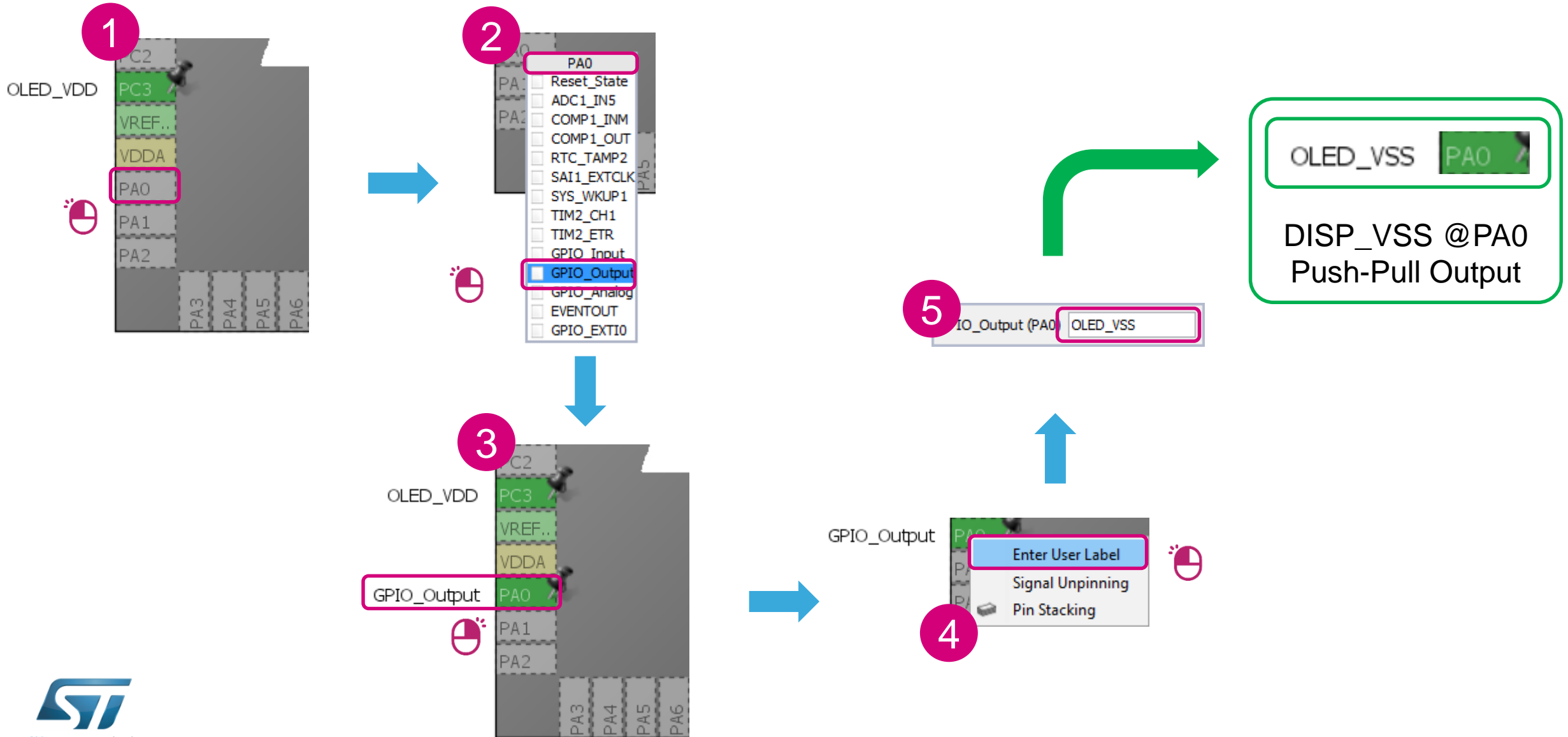
Figure 13-5 : I²C interface Timing characteristics



Add pin for OLED display VDD



Add pin for OLED display VSS



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 (Ctrl+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

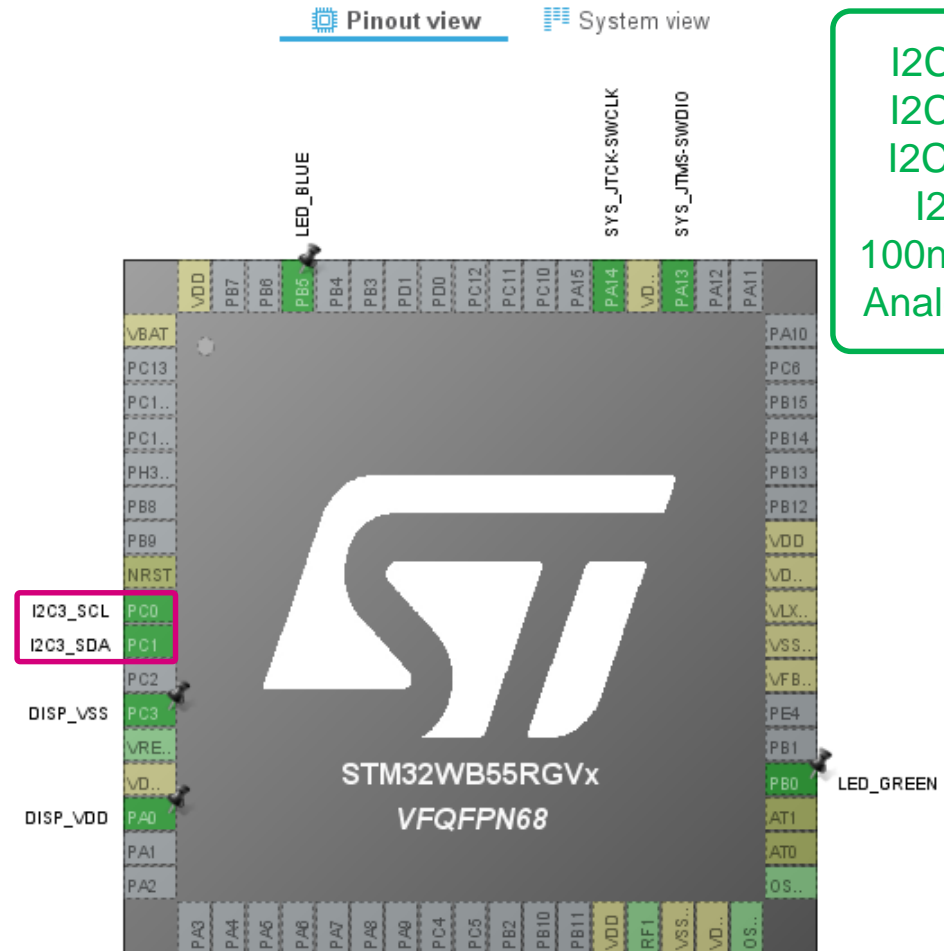
Clock No Stretc... Disabled

General Call Ad... Disabled

Primary Address... 7-bit

Dual Address A... Disabled

Primary slave ad... 0



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

Pinout & Configuration

Additional Softwares

Pinout

Options

Categories A->Z

- System Core
- Analog
- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware

I2C1

✓ I2C3

IRTIM

LPUART1

QUADSPI

RF

SPI1

SPI2

✓ USART1

USB

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 :

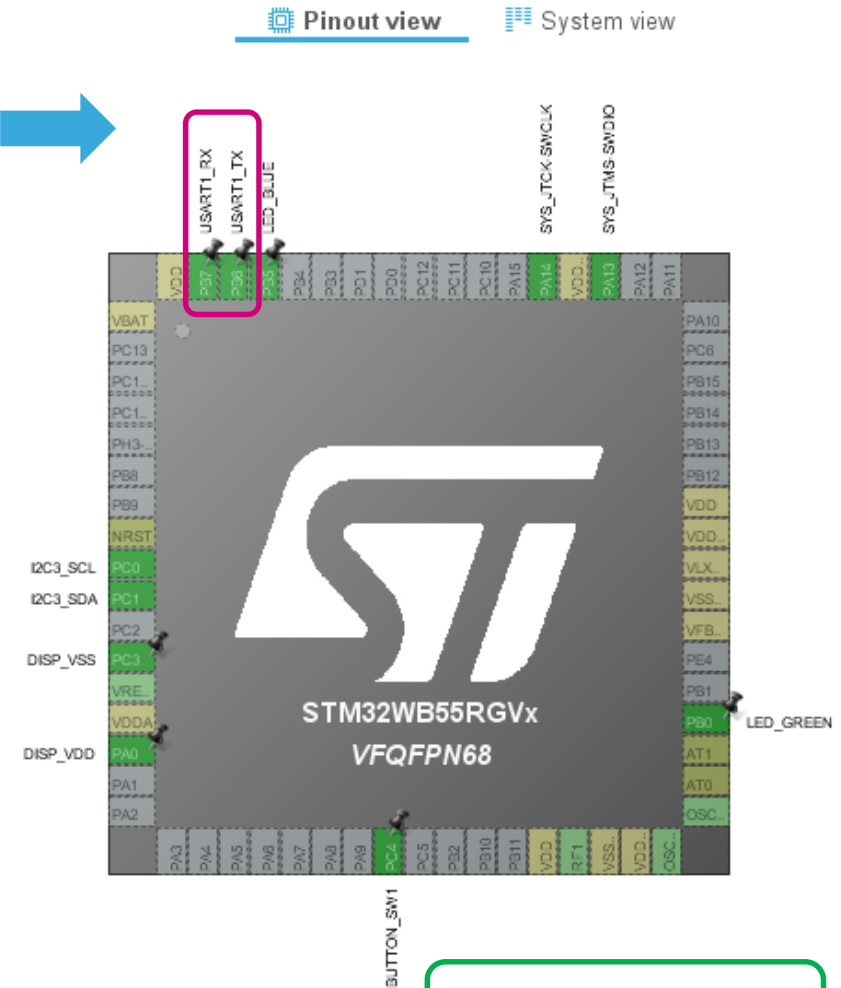
Search (Ctrl+F)

Basic Parameters

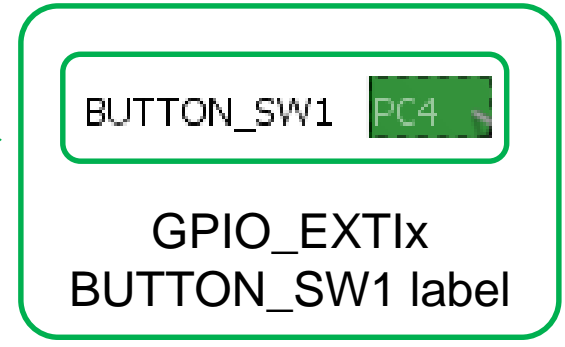
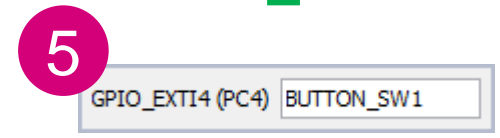
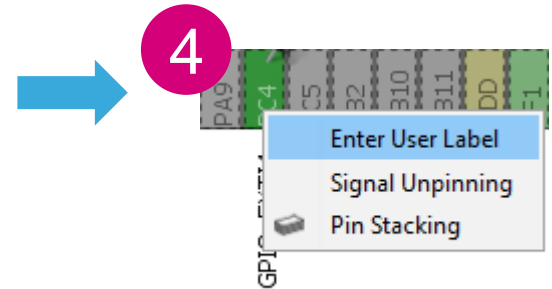
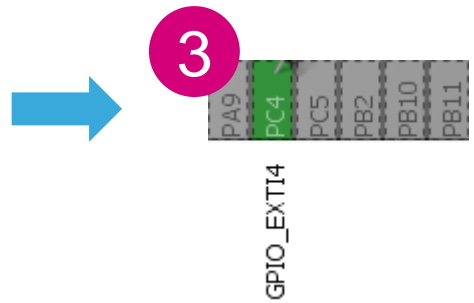
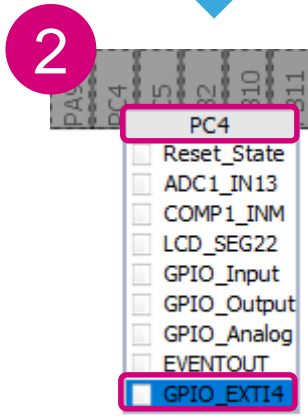
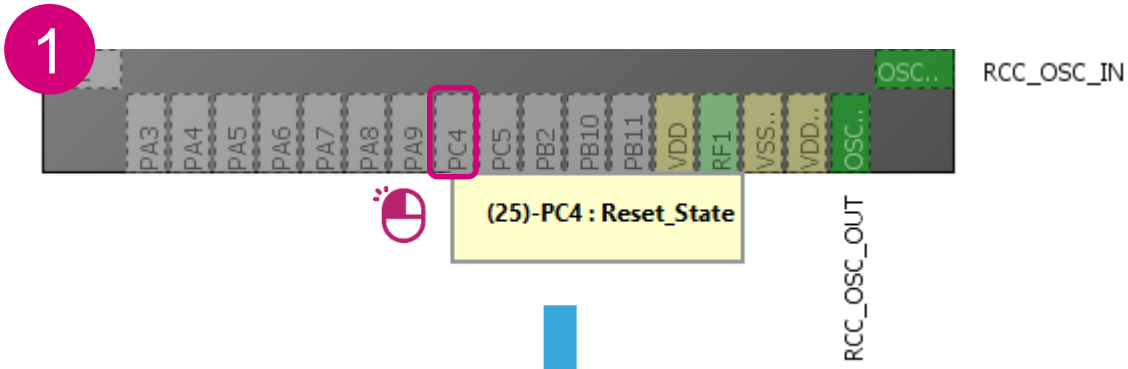
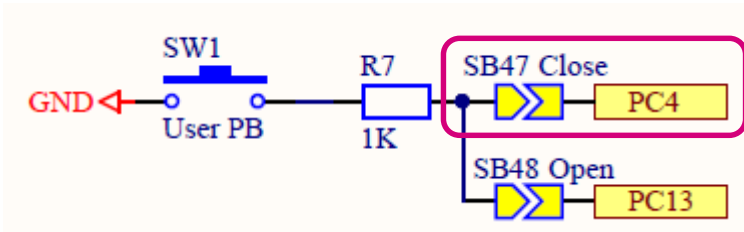
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters

Advanced Features



Add the SW1 Button EXTI input



Configure Button SW1 EXTI input pin

Pinout & Configuration

Additional Softwares Pinout

Options GPIO Mode and Configuration

Categories A->Z

System Core

- DMA
- GPIO**
- HSEM
- IWDG
- NVIC
- RCC
- ▲ SYS
- ▲ TSC
- WWDG

Analog >

Configuration

Group By Peripherals

GPIO
 I2C3
 SYS
 USART1
 NVIC

Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin ...	Signa...	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	Fast Mode	User Label	Modified
PA0	n/a	Low	Output Push Pull	No pull-up and no pull-down	Very High	n/a	DISP_VDD	<input checked="" type="checkbox"/>
PB0	n/a	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	LED_GREEN	<input checked="" type="checkbox"/>
PB5	n/a	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	LED_BLUE	<input checked="" type="checkbox"/>
PC3	n/a	Low	Output Push Pull	No pull-up and no pull-down	Very High	n/a	DISP_VSS	<input checked="" type="checkbox"/>
PC4	n/a	n/a	External Interrupt Mode with Falling edge trigger detection	Pull-up	n/a	n/a	BUTTON_SW1	<input checked="" type="checkbox"/>

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

Enable EXTI line4 interrupt

Pinout & Configuration

Additional Softwares Pinout

Options GPIO Mode and Configuration

Categories A->Z Configuration

Group By Peripherals

GPIO
 I2C3
 SYS
 USART1
 NVIC

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line4 interrupt	<input checked="" type="checkbox"/>	0	0

System Core

- DMA
- GPIO**
- HSEM
- IWDG
- NVIC
- RCC
- ▲ SYS
- ▲ TSC
- WWDG

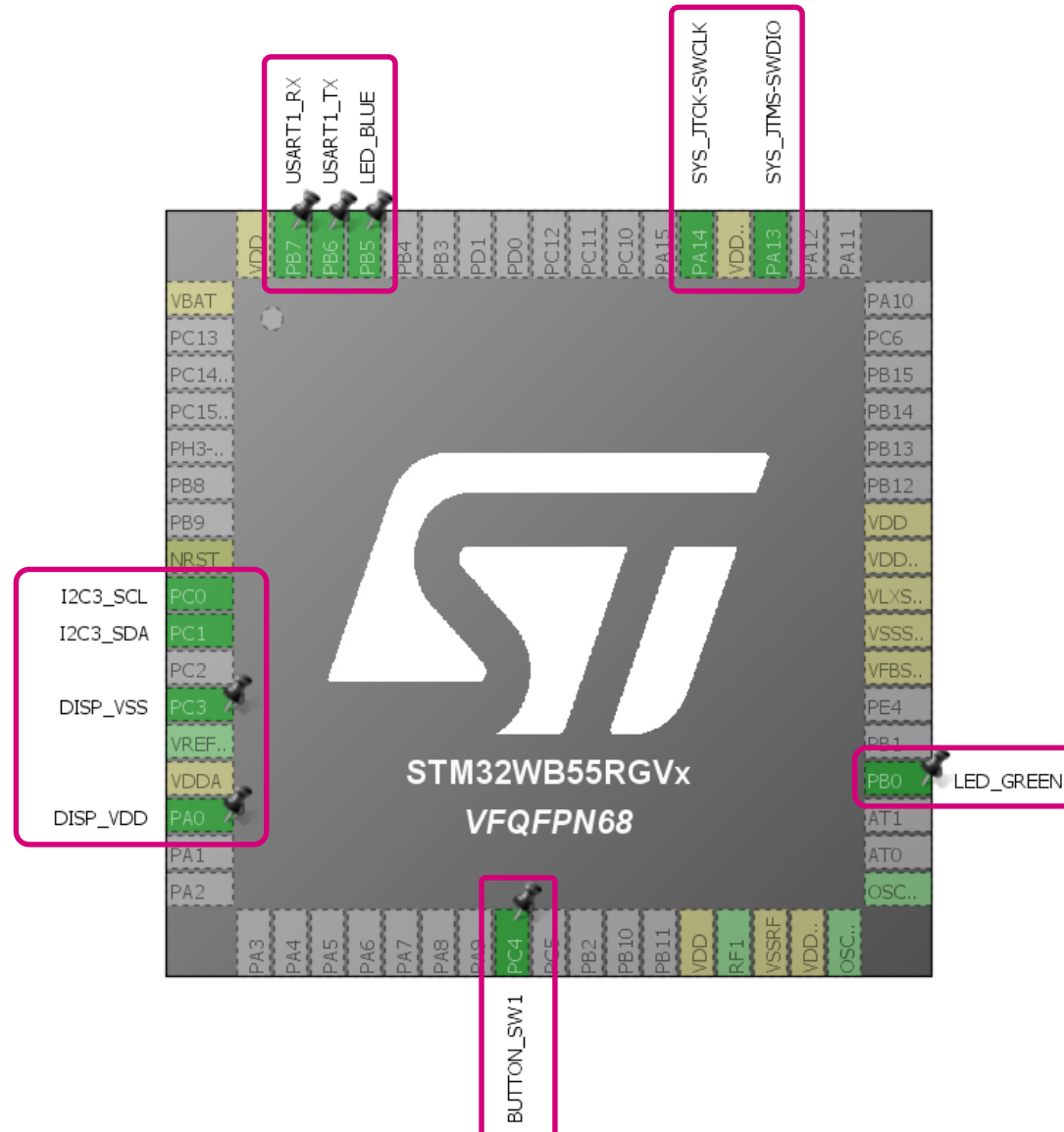
Analog >

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



Continue with Pinout Configuration

The screenshot shows the STM32CubeMX software interface. The title bar reads "MX STM32CubeMX HandsOn_2.ioc: STM32WB55RGVx". The menu bar includes "File", "Window", and "Help". The breadcrumb trail shows "Home > STM32WB55RGVx > HandsOn_2.ioc - Pinout & Configuration". A "GENERATE CODE" button is visible in the top right. The main interface has a sidebar on the left with "Options" and "Categories" (A->Z). The sidebar lists categories: System Core, Analog, Timers, Connectivity, Multimedia, Security, Computing, and Middleware. The main area is titled "Pinout & Configuration" and shows a "Pinout" view of the STM32WB55RGVx VFQFPN68 package. The package is a 68-pin VFQFPN68. The pins are labeled with their functions: USART1_TX, USART1_TX, LED_BLUE, LED_GREEN, SYS_TCK-SWCLK, SYS_TMS-SWDIO, VBAT, PC13, PC1, PC1, PH3, PB9, PB9, NRST, I2C3_SCL, I2C3_SDA, DISP_VSS, VRE, VDDA, DISP_VDD, PA1, PA2, PA3, PA4, PA5, PA6, PA7, PA8, PA9, PA10, PA11, PA12, PA13, PA14, PA15, PC11, PC12, PC10, PA15, PA15, VDD, VDD, PA12, PA11, PA10, PC6, PB15, PB14, PB13, PB12, VDD, VDD, VLX, VSS, VFB, PE4, PB1, PA0, AT1, AT0, OSC, OSC.



Add LSE 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) Crystal/Ceramic Resonator ▼

Master Clock Output

LSCO Clock Output

SAI1 Extern CLock

CRS SYNC Disable ▼

Pinout view System view

RCC_OSC32_IN PC1

RCC_OSC32_OUT PC1

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

STM32WB55RGVx
VQFP68

STM32WB logo



Enable Hardware Semaphores (HSEM)

Pinout & Configuration

Additional Softwares

HSEM Mode and Configuration

Mode
<input checked="" type="checkbox"/> Activated

Options

Categories A->Z

System Core

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

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

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 >
- Mid

RF Mode and Configuration

Mode

Activate RF1

Pinout view System view

Connectivity → RF
Activate RF1
RF_RF1 pin assigned

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



Enable RTC

114

Pinout & Configuration

Additional Softwares

Options

Categories A->Z

System Core >

Analog >

Timers

- LPTIM1
- LPTIM2
- RTC**
- TIM1
- TIM2
- TIM16
- TIM17

RTC Mode and Configuration

Mode

- Activate Clock Source
- Activate Calendar
- Alarm A: Disable
- Alarm B: Disable
- Timestamp
- WakeUp: Disable
- Tamper 1
- Tamper 2
- Tamper 3
- Calibration: Disable
- Reference clock detection

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.



Enable STM32_WPAN BLE Middleware

Pinout & Configuration

Additional Softwares

Options

Categories A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware** ▾

FATFS

FREERTOS

✓ STM32_WPAN

TOUCHSENSING

USB_DEVICE

STM32_WPAN Mode and Configuration

Mode

- BLE**
- THREAD*

Middleware → STM32_WPAN BLE



Configure STM32_WPAN BLE Middleware

Pinout & Configuration

Additional Softwares Pinout

Options

Categories A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware ▾
 - FATFS
 - FREERTOS
 - STM32_WPAN
 - TOUCHSENSING
 - USB_DEVICE

STM32_WPAN Mode and Configuration

Configuration

Reset Configuration

- Parameter Settings
- User Constants
- BLE Applications and Services
- Configuration

Configure the below parameters :

Search (Ctrl+F) ⏪ ⏩ ⓘ

- BLE Application Type
 - BLE Application Type Server profile
- Server Mode
 - BT SIG Beacon Disabled
 - BT SIG Blood Pressure Sensor Disabled
 - BT SIG Health Thermometer Sensor Disabled
 - BT SIG Heart Rate Sensor Disabled
 - Custom P2P Server Enabled
 - Custom Template Disabled
- > BLE Services Configuration
- > P2P Service
 - P2P_SERVER_NUMBER P2P_SERVER1
- > Local Name

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

Pinout & Configuration

Additional Softwares Pinout

Options

Categories A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware
 - FATFS
 - FREERTOS
 - STM32_WPAN**
 - TOUCHSENSING
 - USB_DEVICE

STM32_WPAN Mode and Configuration

Configuration

Reset Configuration

- Parameter Settings
- User Constants
- BLE Applications and Services
- Configuration

Configure the below parameters :

- BLE Application Type
 - BLE Application Type: Server profile
- Server Mode
 - BT SIG Beacon: Disabled
 - BT SIG Blood Pressure Sensor: Disabled
 - BT SIG Health Thermometer Sensor: Disabled
 - BT SIG Heart Rate Sensor: Disabled
 - Custom P2P Server: Enabled
 - Custom Template: Disabled
- BLE Services Configuration
- P2P Service
 - P2P_SERVER_NUMBER: P2P_SERVER1
- Local Name
 - LOCAL_NAME: XX-NODE

LOCAL_NAME changed according to your happy number

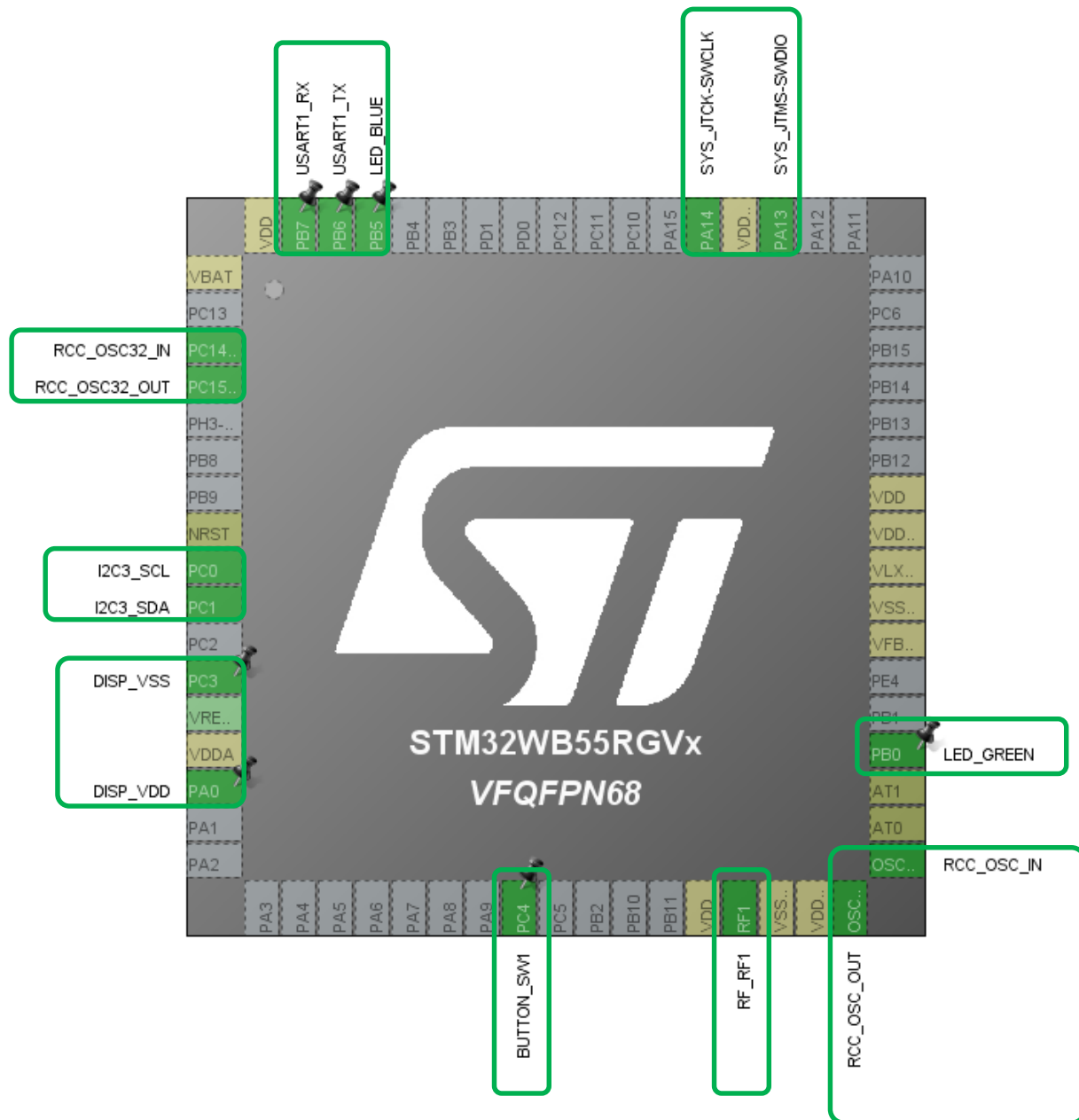
Max number of characters is set to 7 to be aligned with overall length of advertising data used by the generated code.

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



System Core

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

Timers

- LPTIM1
- LPTIM2
- RTC
- TIM1
- TIM2
- TIM16
- TIM17

Connectivity

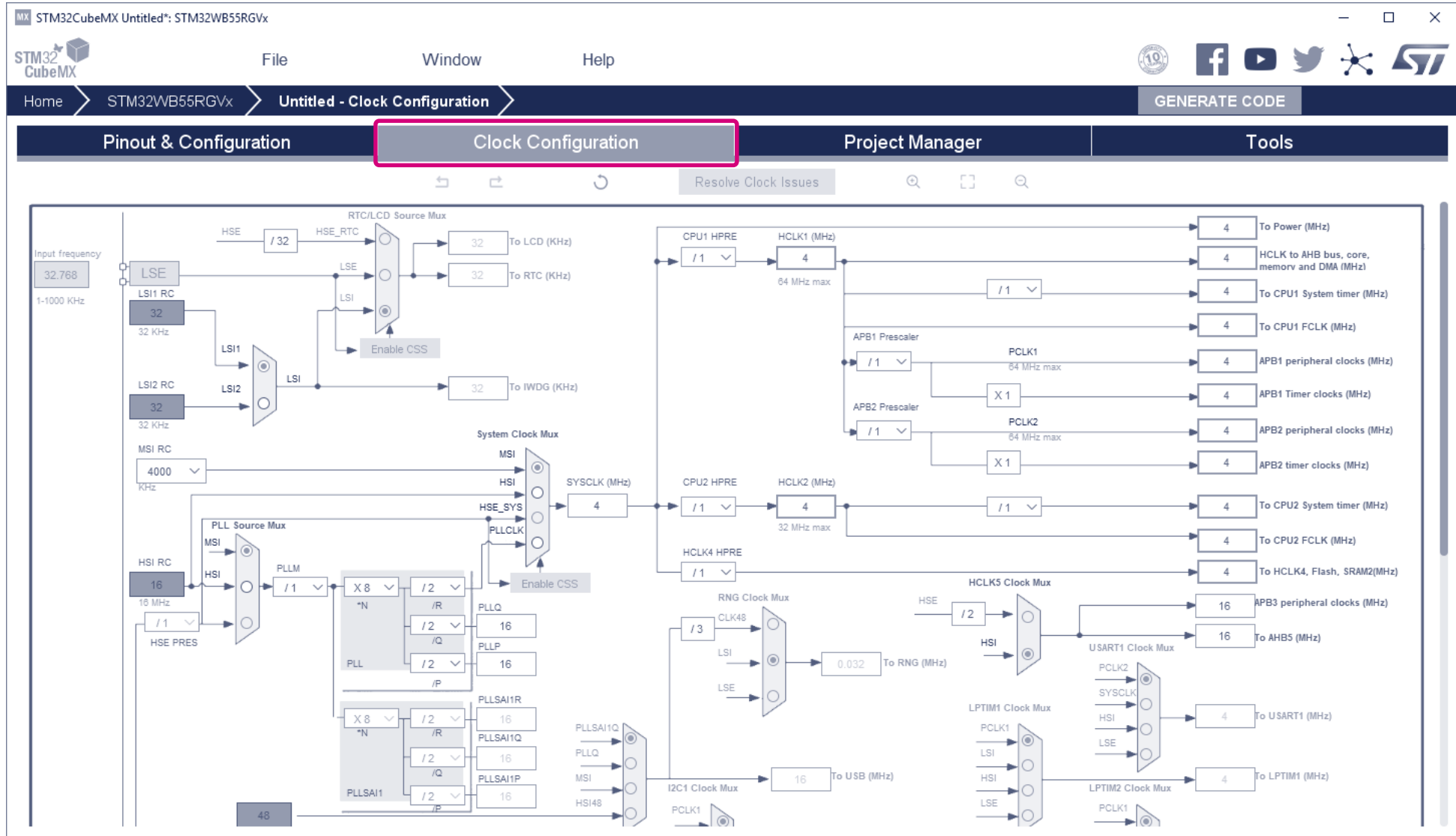
- I2C1
- I2C3
- IRTIM
- LPUART1
- QUADSPI
- RF
- SPI1
- SPI2
- USART1
- USB

Middleware

- FATFS
- FREERTOS
- STM32_WPAN
- TOUCHSENS...
- USB_DEVICE



Switch to clock configuration





Clock issue solver

121

⊗ Clock Configuration

Clock configuration ×

? Do you want to run automatic clock issues solver ?

Otherwise you can do it later by clicking on button "Resolve Clock Issues"

Do not show this message again.

Remember my decision for next projects.

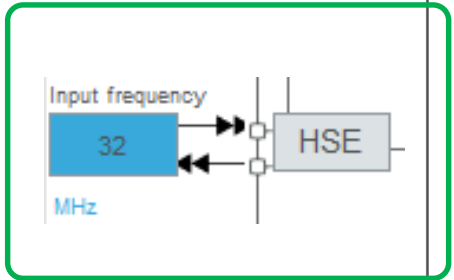
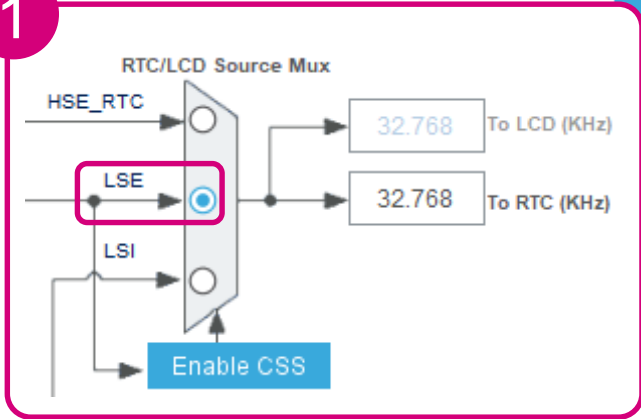
The clocks requirements for select peripherals (e.g. 32MHz for RF) are not fulfilled, click on Yes, when the Clock issue solver dialog pops up



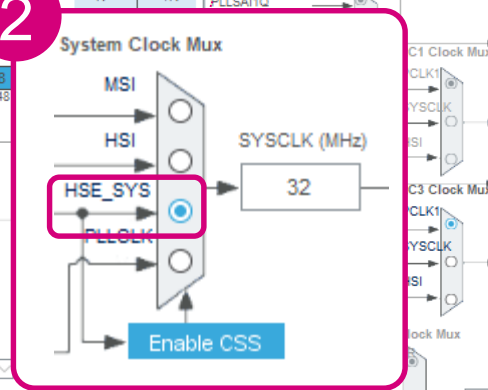
Clock configuration

1

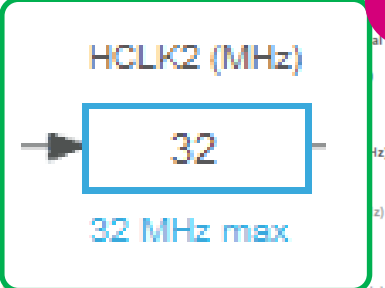
Clock Configuration



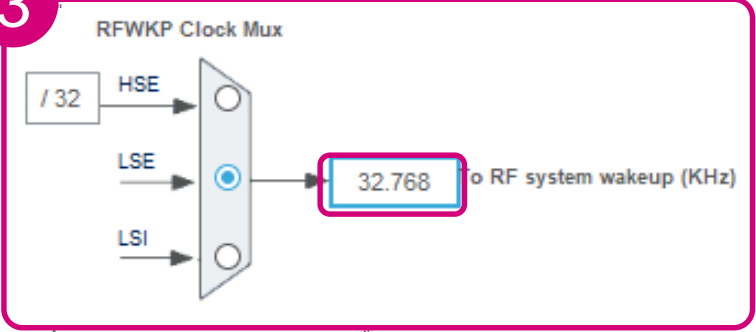
2



3



HSE 32MHz
 SYSCLK HSE_SYS
 → HCLKx 32MHz
 LSE 32.768kHz
 RTC from LSE
 RFWKP from LSE





Finalize the project settings

The screenshot shows the STM32CubeMX Project Manager window. The 'Project Manager' tab is selected and highlighted with a pink border. The window is divided into four main sections: Pinout & Configuration, Clock Configuration, Project Manager, and Tools. The Project Manager section is further divided into Project, Code Generator, and Advanced Settings.

Project Settings:

- Project Name: HandsOn_1
- Project Location: C:\STM32WB_workshop
- Application Structure: Basic
- Toolchain Folder Location: C:\STM32WB_workshop\HandsOn_1\
- Toolchain / IDE: TrueSTUDIO

Linker Settings:

- Minimum Heap Size: 0x200
- Minimum Stack Size: 0x400

Mcu and Firmware Package:

- Mcu Reference: STM32WB55RGVx
- Firmware Package Name and Version: STM32Cube FW_WB V1.0.0
- Use Default Firmware Location:

MCUs Selection Table:

MCUs Selection	Output	Series	Lines	Mcu	Package	Required Peripherals
<input checked="" type="radio"/>		STM32WB	STM32WBx5	STM32WB55RGVx	VFQFPN68	None



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

Project settings

Project Manager

1 Project Settings

Project Name: HandsOn_2

Project Location: C:\STM32WB_workshop\HandsOns

2 Application Structure: Basic

Toolchain Folder Location: C:\STM32WB_workshop\HandsOns\HandsOn_2\

3 Toolchain / IDE: TrueSTUDIO

Linker Settings: Minimum Heap Size: 0x200, Minimum Stack Size: 0x400

4 Mcu and Firmware Package: STM32WB55RGVx

Firmware Package Name and Version: STM32Cube FW_WB V1.0.0

Use Default Firmware Location:

Keep all the other options in default state!

HandsOn_2

C:\STM32WB_workshop\HandsOns

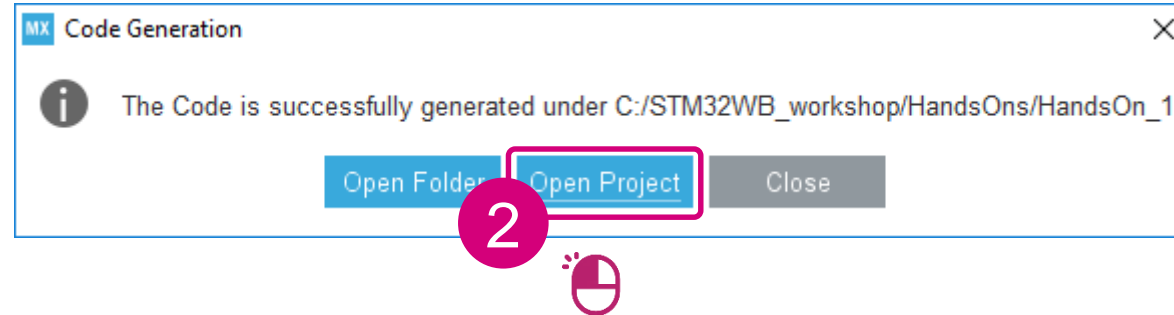
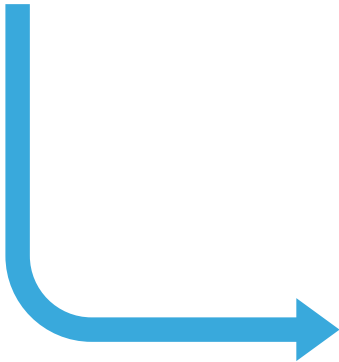
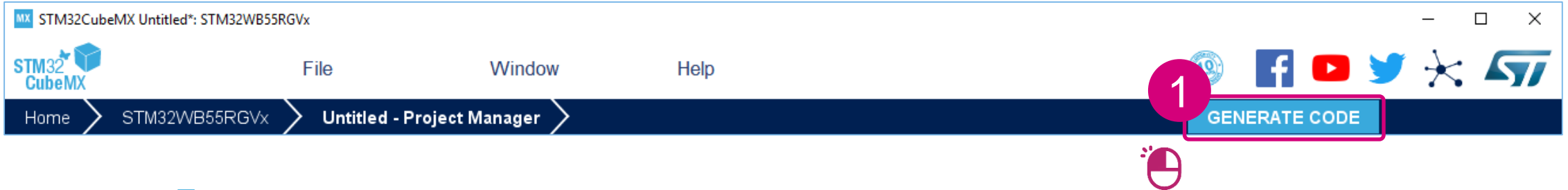
TrueSTUDIO

STM32Cube_FW_WB_V1.0.0



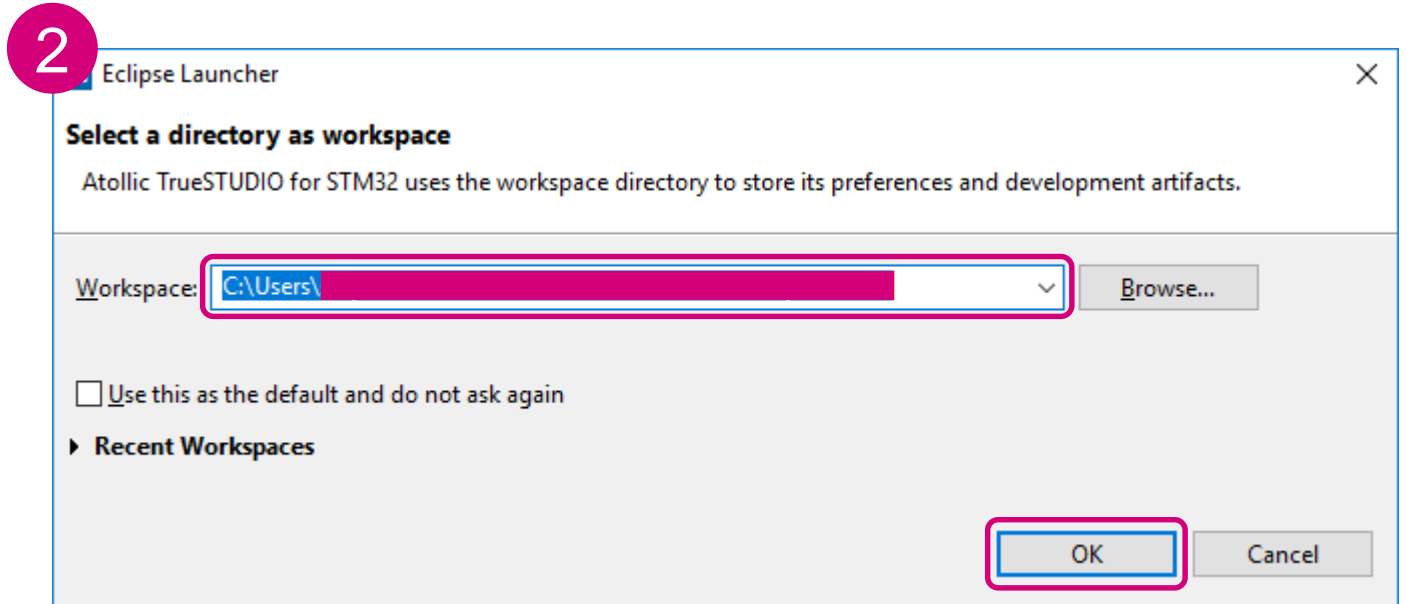


Generate the code



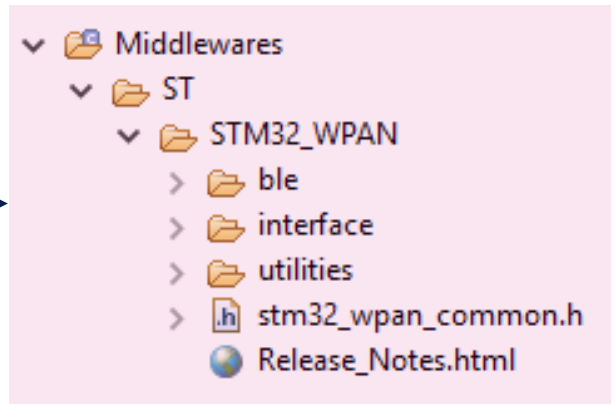
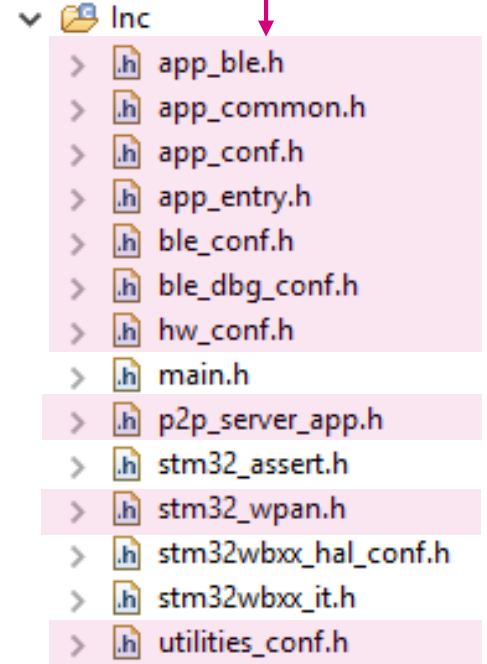
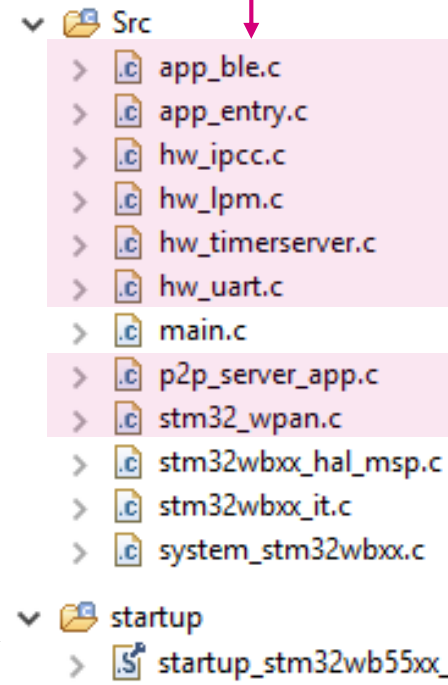
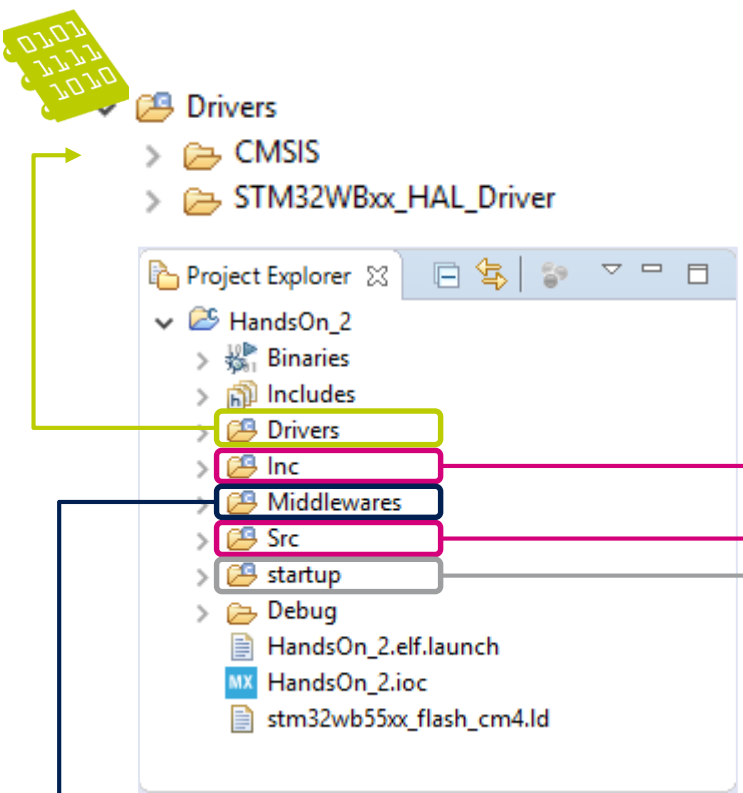
Atollic TrueStudio project opening

126



Select the workspace

Check out the project tree



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



STM32 system blocks and IPs initialization

128

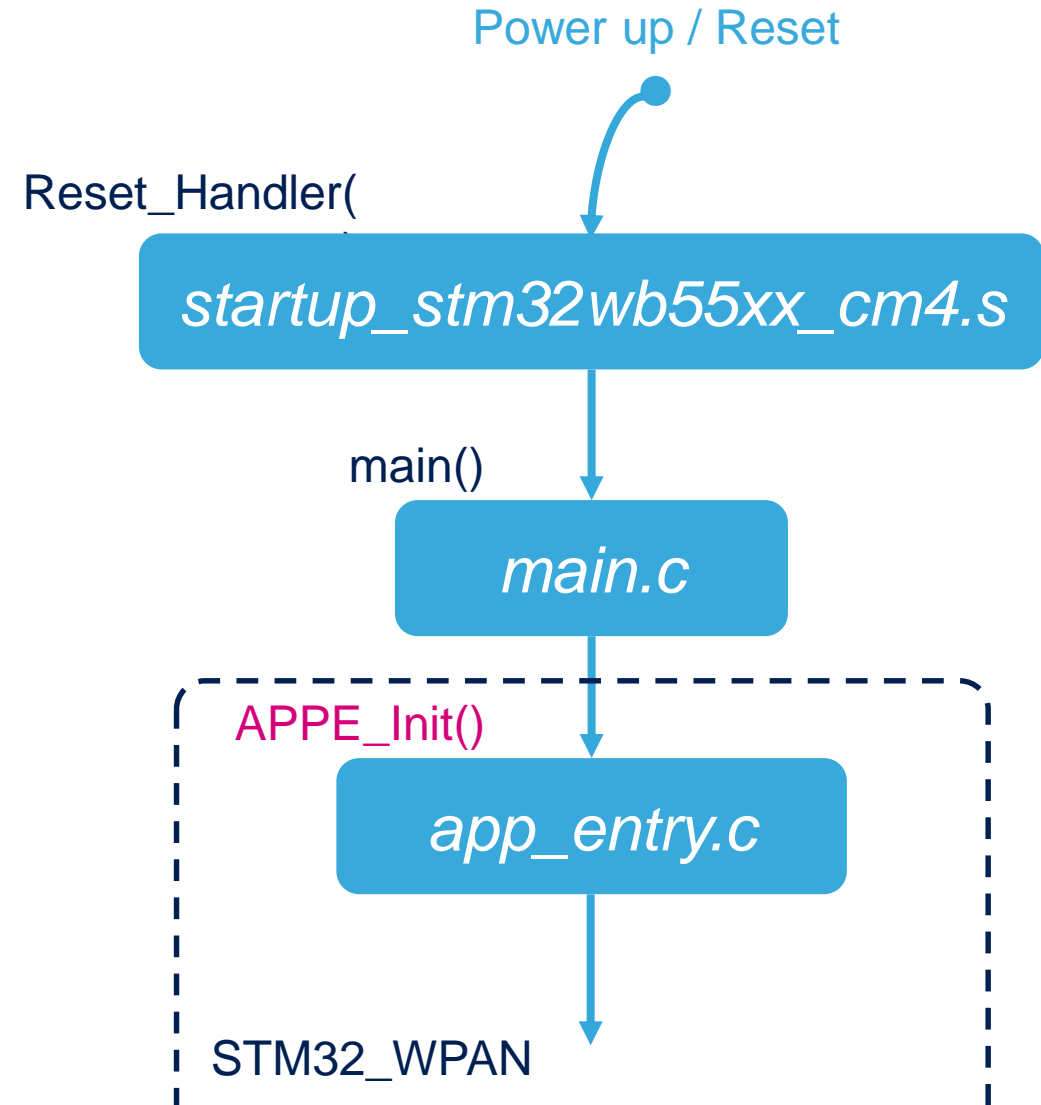
- Reset_Handler

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

→ main()

- main()

- MCU HW initialization
 - RCC (clock), GPIO, RTC, I2C3,...
- APPE_Init()





stm32wbxx_hal_msp.c

Add the HSE tuning

129

ADD

1

```
user section { Includes @Line ~24 }
/* USER CODE BEGIN Includes */
#include "otp.h"
/* USER CODE END Includes */
```



STEP1_Add_HSE_tuning.txt

2

```
HAL_MspInit(...) user section { MspInit 0 @Line ~67}
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



stm32wbxx_it.c

Add STM32_WPAN ISRs

130



STEP2_Add_STM32_WPAN_ISR.txt

ADD

1

```
/* 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 */
```

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



Typical simple application architecture

131

```
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

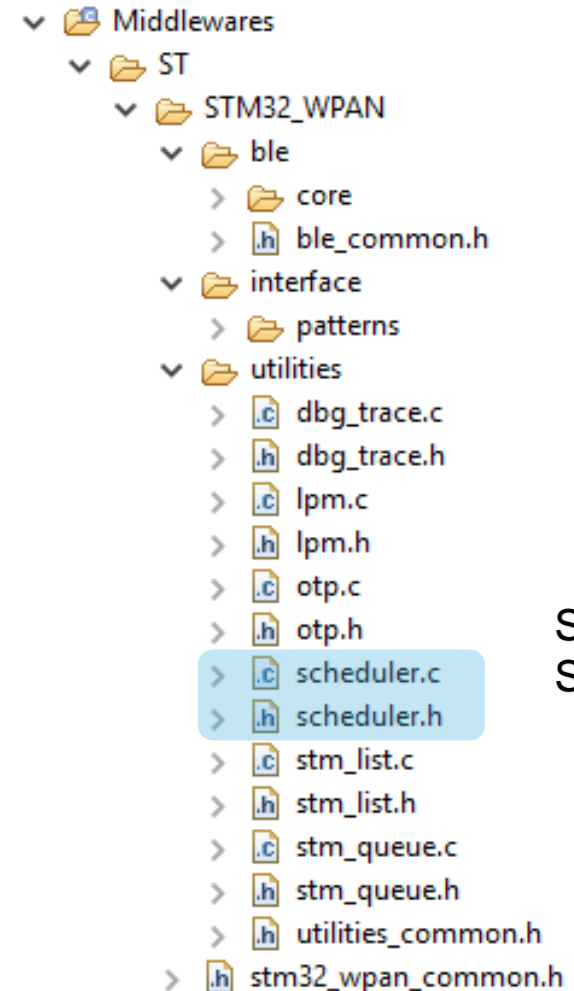


Simple Scheduler

132

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

133

- 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

134

main.c

```
user section { Includes @Line ~40 }
/* USER CODE BEGIN Includes */
1 #include "scheduler.h"
/* USER CODE END Includes */
...
user section { 3 @Line ~116 }
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

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

ADD

Not generated by STM32CubeMX yet
when STM32_WPAN is in use.



STEP3_Add_scheduler_call.txt



OLED display functionality

STEP4_Add_OLED_init.txt

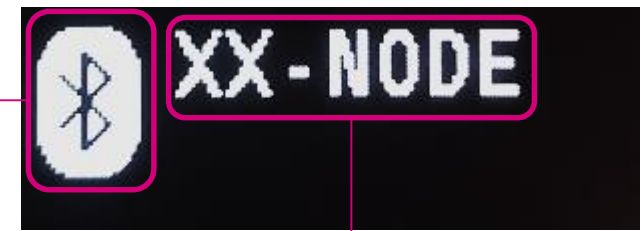
Display the Device name

app_ble.c

```
user section { Includes @Line ~40 }
/* USER CODE BEGIN Includes */
1 #include "hal_lcd.h"
/* USER CODE END Includes */
...
user section { APP_BLE_Init_1 @Line ~381 }
2 /* 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 */
```

ADD

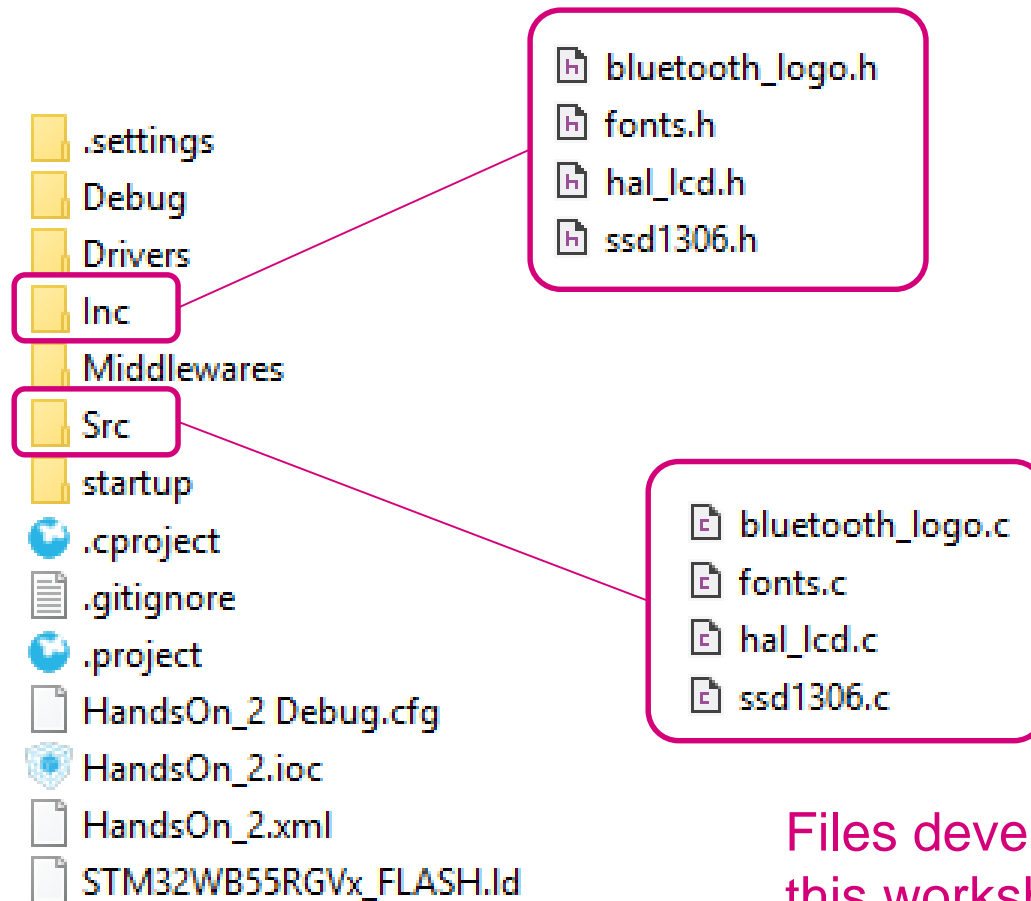
According to the STM32_WPAN architecture, LCD_Init() shall be called from app_entry.c. We will put in app_ble.c just for simplicity.





OLED display driver files

136



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.

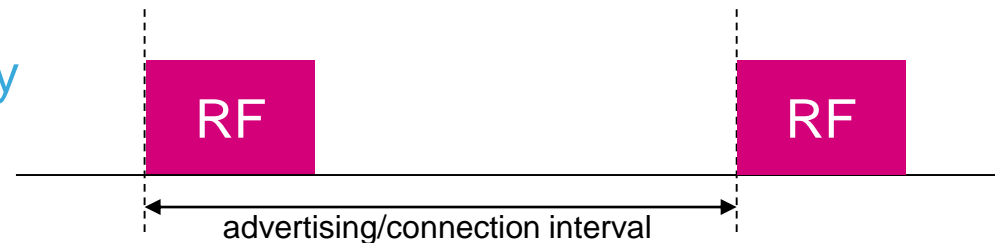


RADIO_ACTIVITY_EVENT

Triggered after every Radio RF activity finishes




Event mask configurable

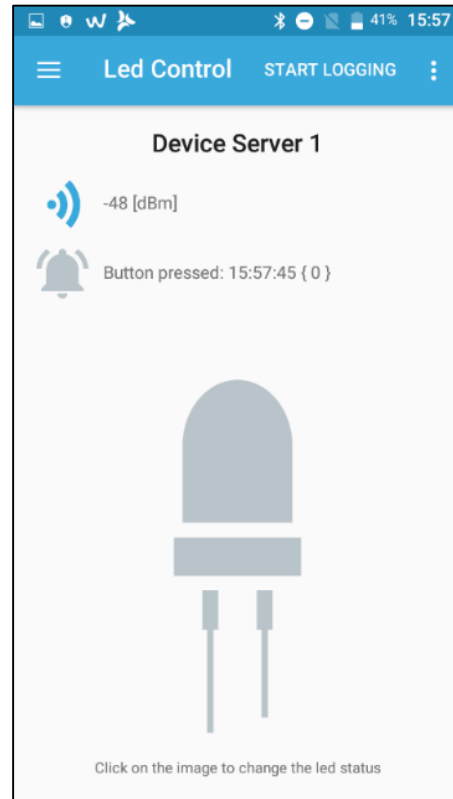
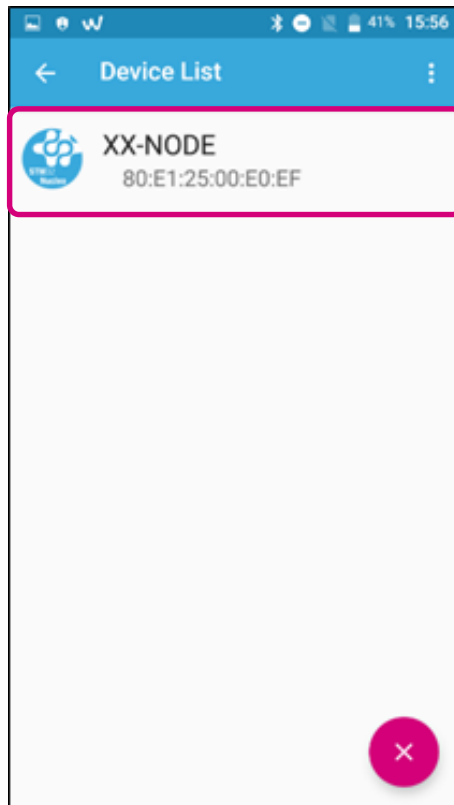
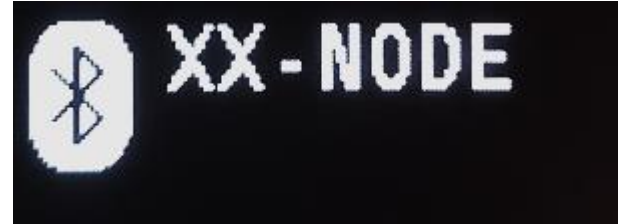
(ADVERTISE, SCAN, CONNECTION)



RADIO_ACTIVITY_EVENT enabled in app_conf.h

Test the functionality

- 1 Build**

or Ctrl+B
- 2 Debug**

F11
- 3 Resume**

F8



GREEN LED blinking period changes when advertising vs. connected

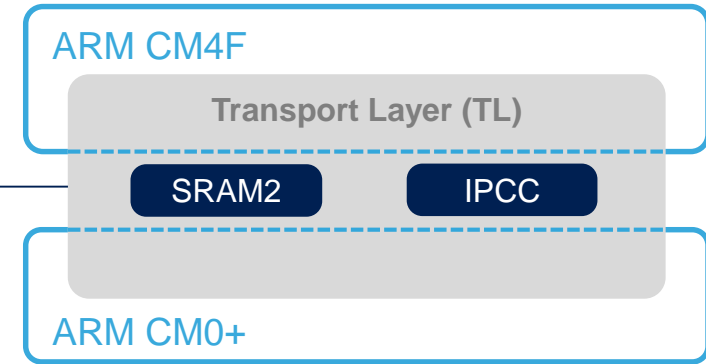
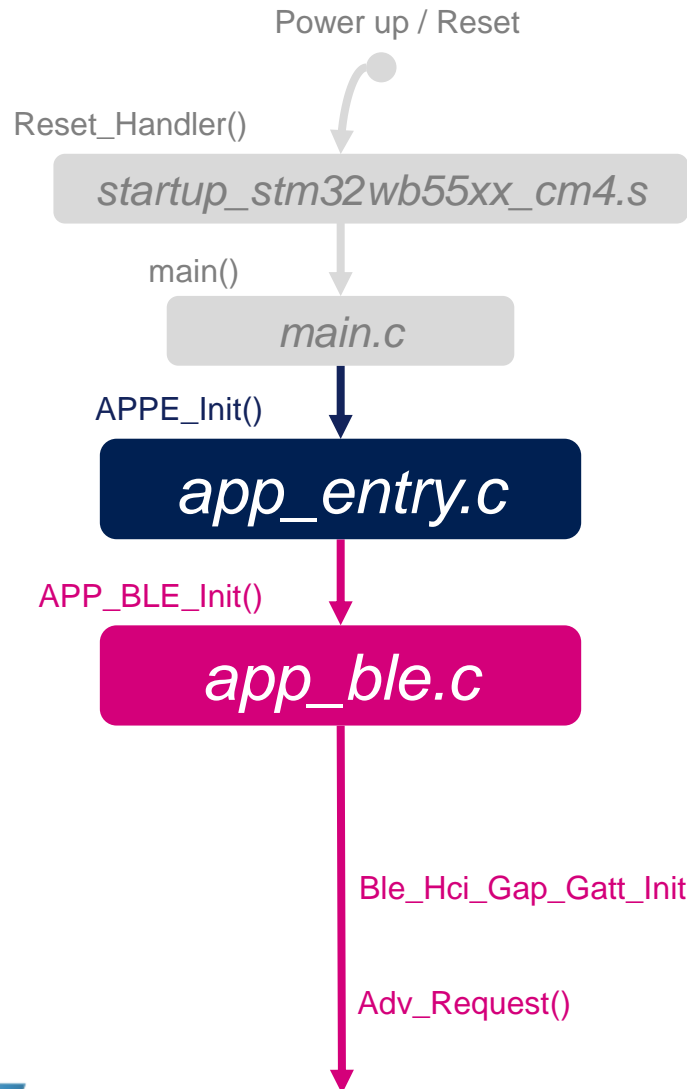
Advertising stops after 60sec if link not established (GREEN LED stops blinking)

Reset for restart

DISCONNECT



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

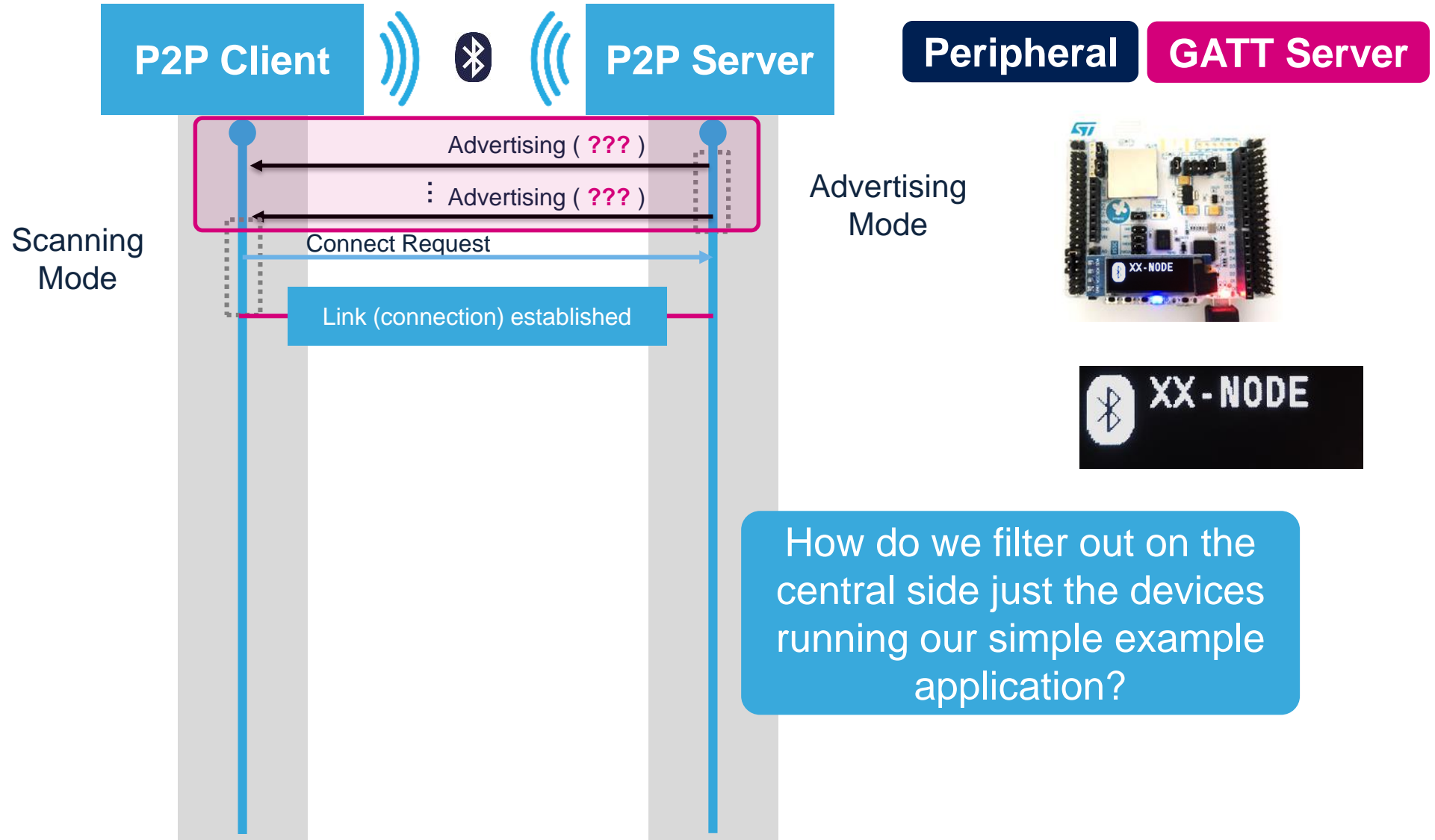
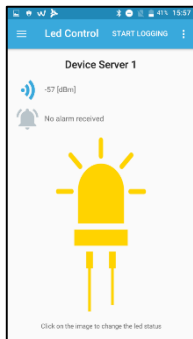
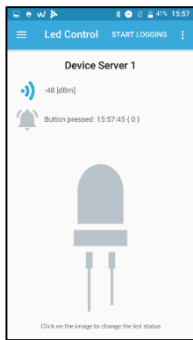
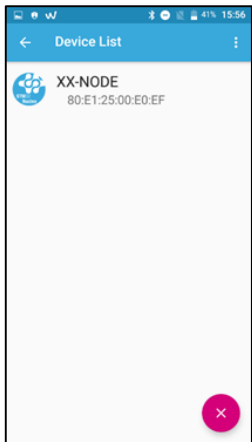
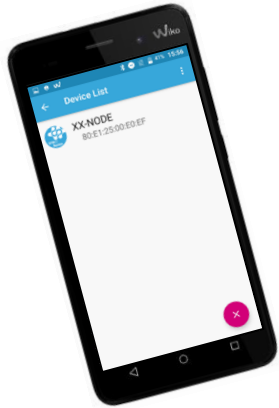
- Set TX Output Power
- GATT init
- GAP init
- Set discoverable (start advertising)

Generated code

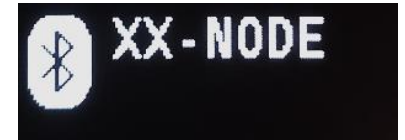


What is running now?

Central GATT Client



Peripheral GATT Server



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



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	0XXXXXXXXXXXXX	0xXX.....XX

Don't care now

Interesting for us

Data PDU

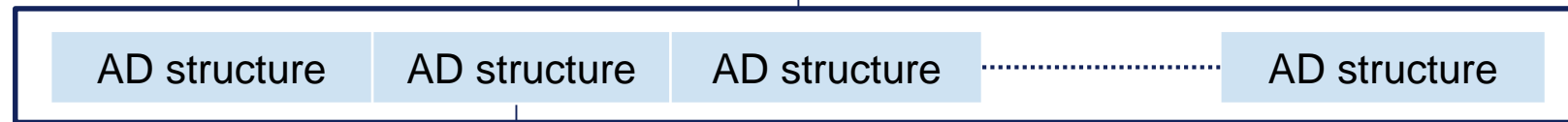
Not under scope now



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 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	0xFFFF.....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' };
```



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(...);`



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	0XXXXXXXXXX	0XXXXXXXXXXXXXX

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 \(@github\)](#)



app_ble.c

Manufacturer specific data

146

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 */  
};
```

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)

```
.....  
/* USER CODE END PV */
```



Advertising data update

147

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

148

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(...);`

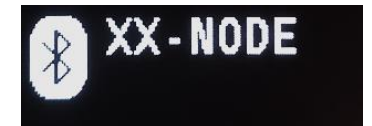
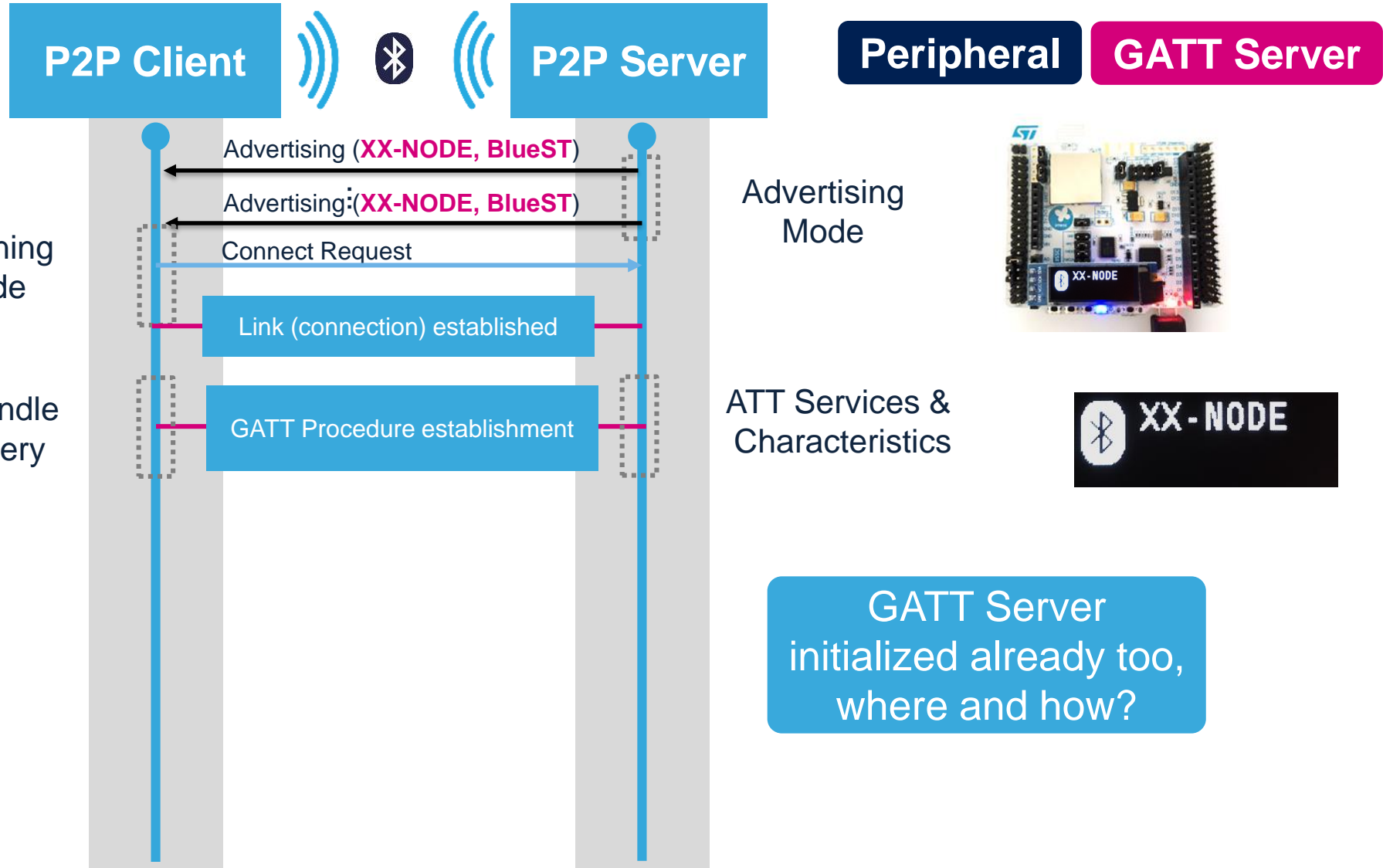
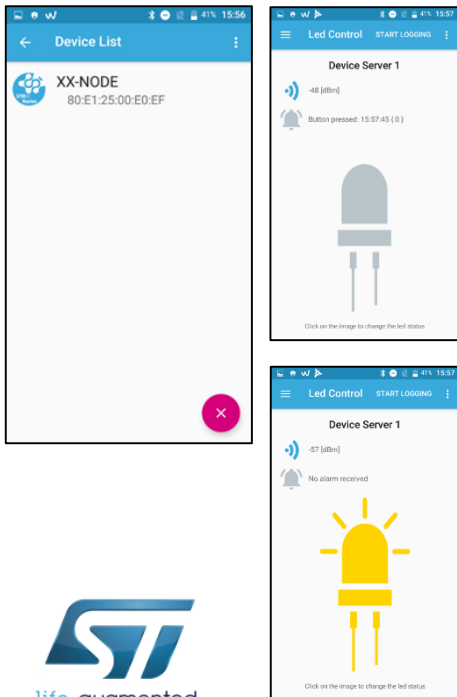
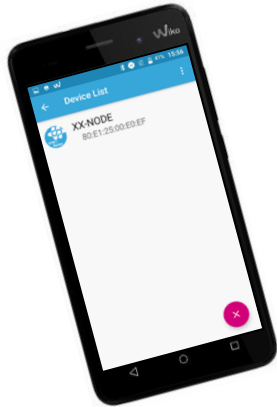
Update advertising data —————→ `aci_gap_update_adv_data(...);`

Stop advertising —————→ `aci_gap_set_non_discoverable(...);`



What is running now?

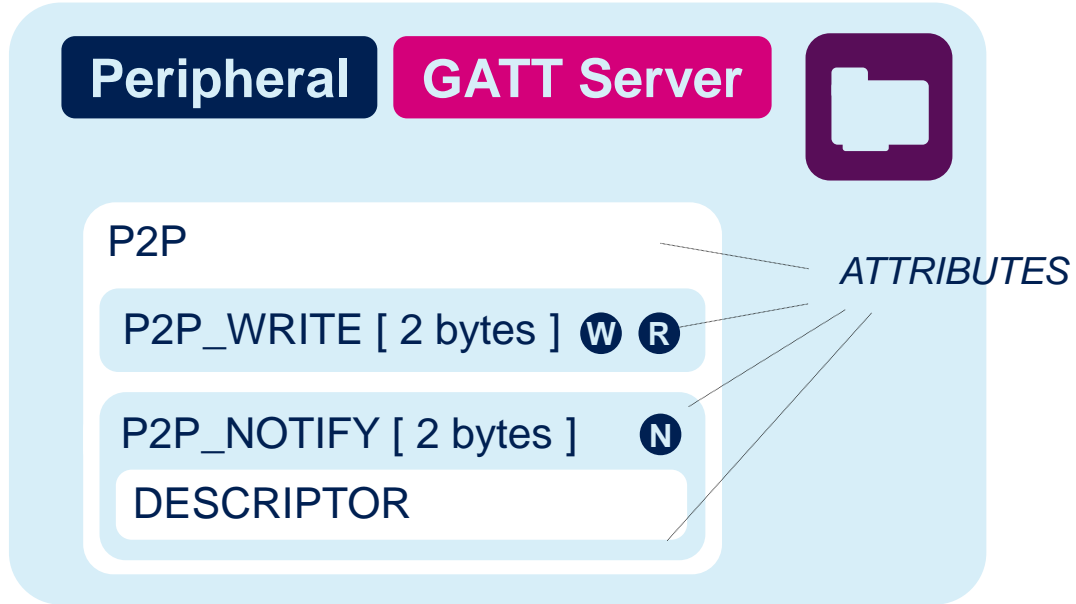
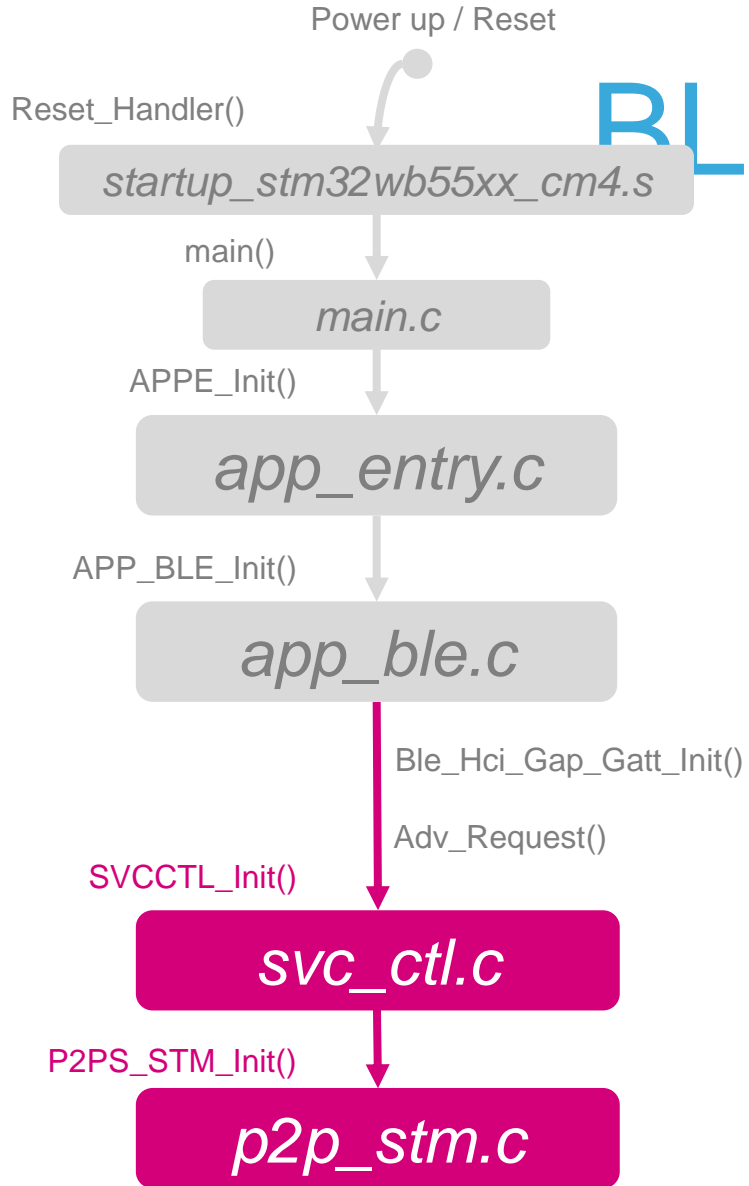
Central **GATT Client**



GATT Server initialized already too, where and how?



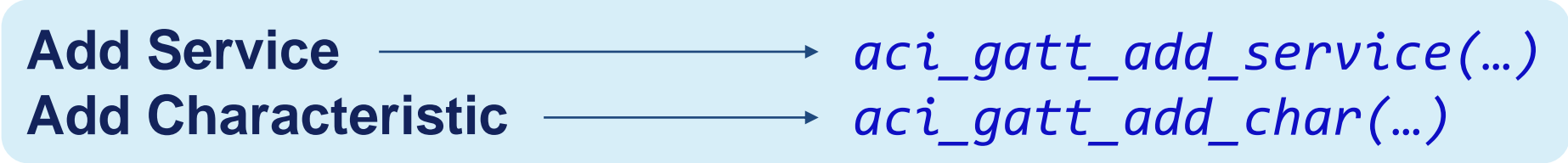
BLE Application Initialization



Add **GATT** services and characteristics

Generated code

Add P2P STM service and characteristics





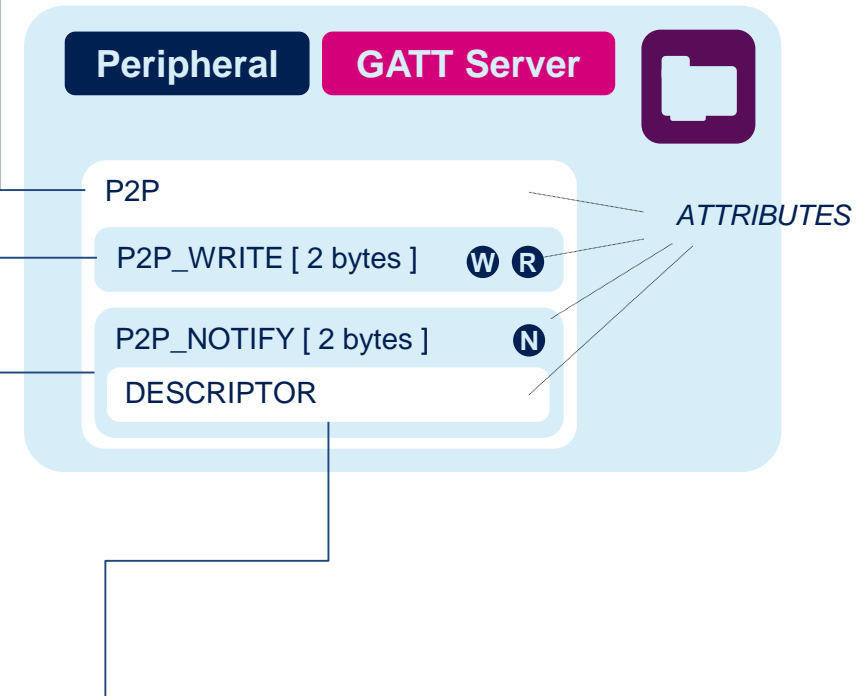
P2P_STM Service overview

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

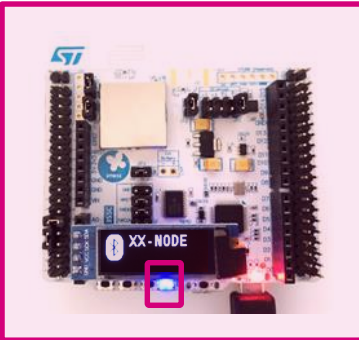
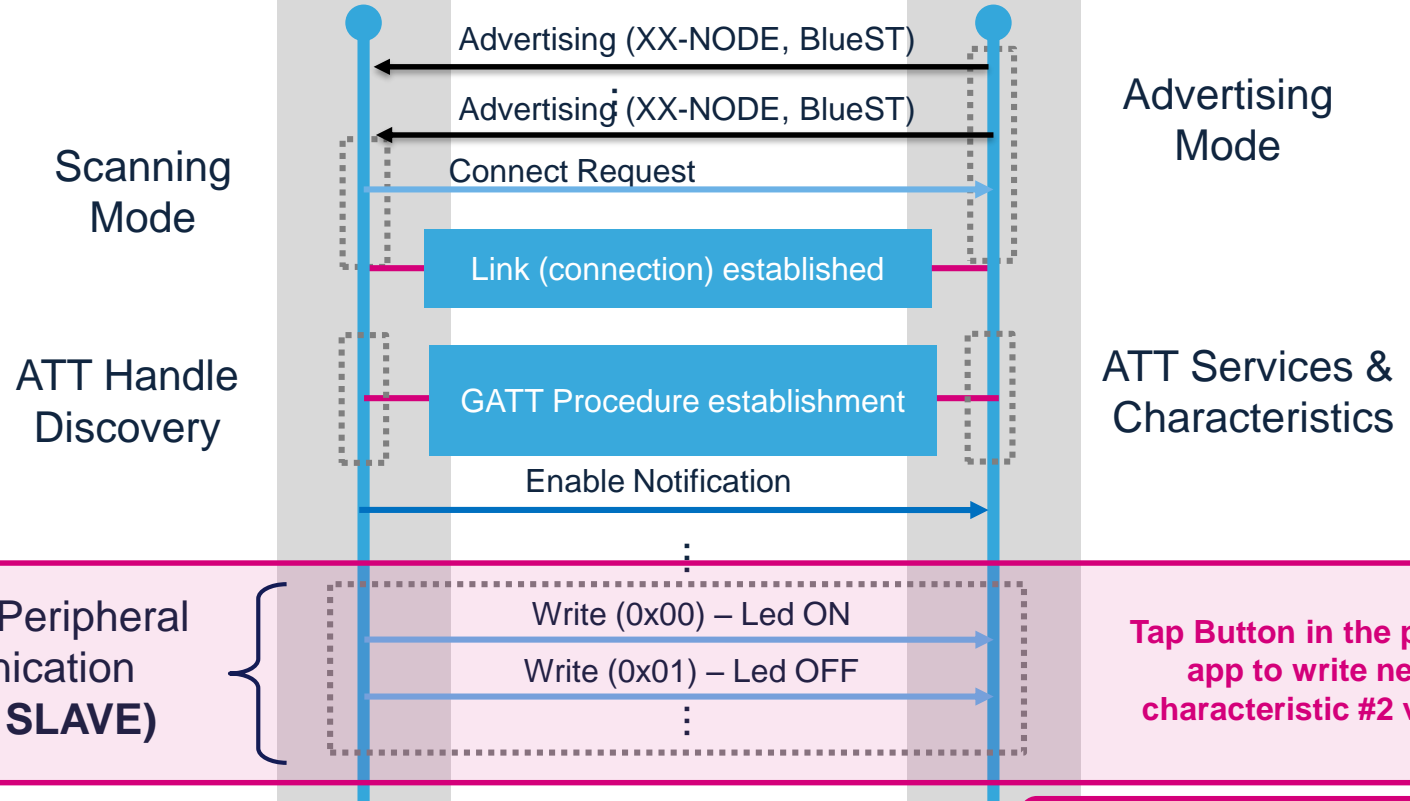
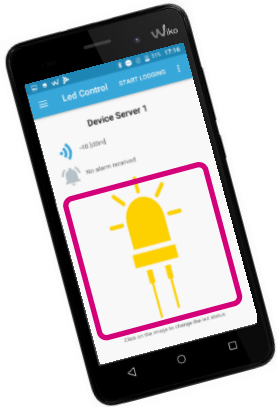
UUID (hex)	2902 (Client Characteristic Configuration)			
Value	Bit	15~2	1 (Indication state)	0 (Notification state)
		Reserved for future use	0 - Indications disabled 1 - Indications enabled	0 - Notifications disabled 1 - Notifications enabled





What to do next?

Central GATT Client



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



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

Attribute 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->DataTransferred.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 */  
    :  
}
```

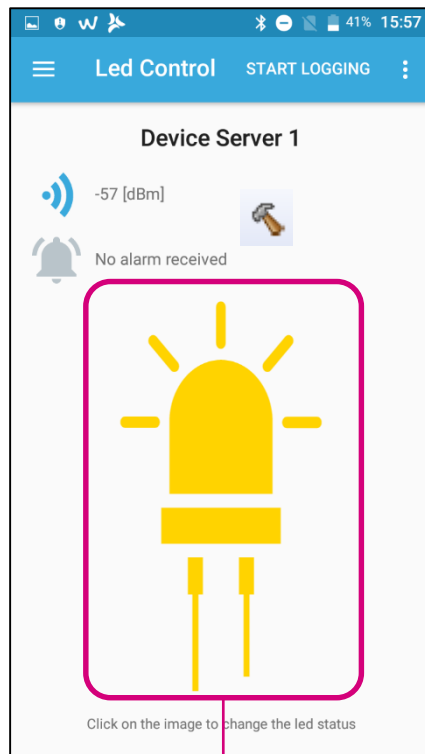
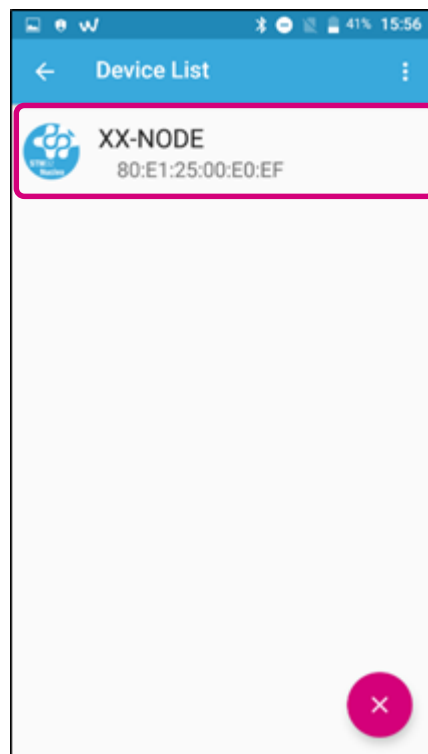
ADD

Turn the blue LED OFF otherwise

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

- 1 Build

or Ctrl+B
- 2 Debug

F11
- 3 Resume

F8

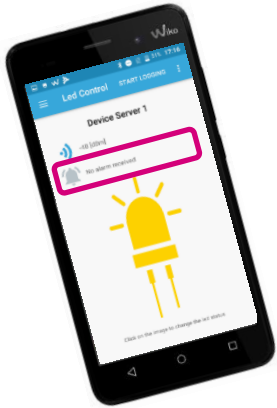
Test the functionality



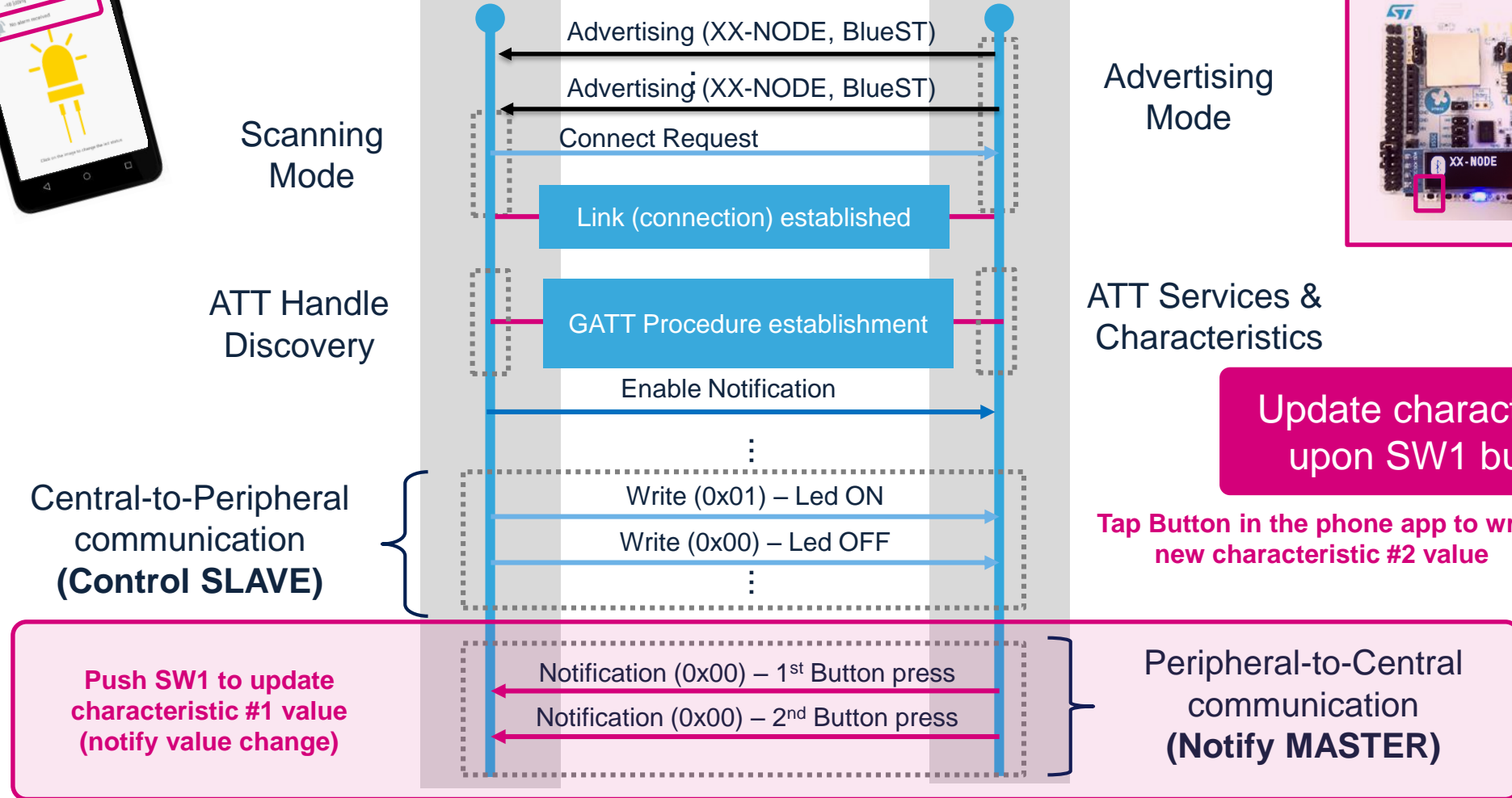
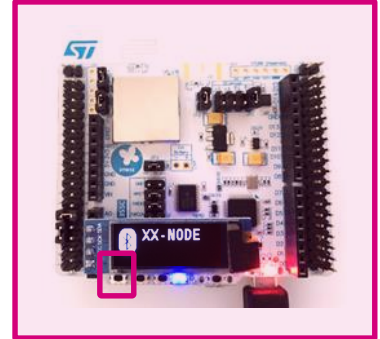


What to do next?

Central GATT Client



Peripheral GATT Server



Scanning Mode

Advertising Mode

ATT Handle Discovery

ATT Services & Characteristics

Central-to-Peripheral communication (Control SLAVE)

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

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

Update characteristic value upon SW1 button press

Peripheral-to-Central communication (Notify MASTER)



Implement Send Notification function

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



STEP7_Add_P2PS_Send_Notification.txt

```
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 */
```

p2p_stm.c

Update Characteristic value → *aci_gatt_update_char_value(...)*



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 */
```

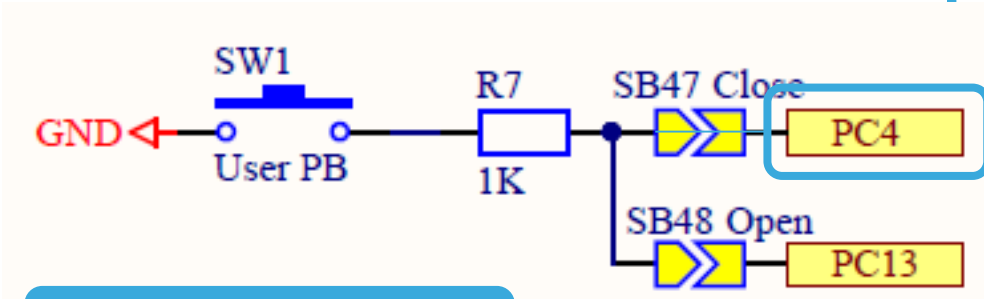
```
}
```

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 */
}
```

GENERATED CODE

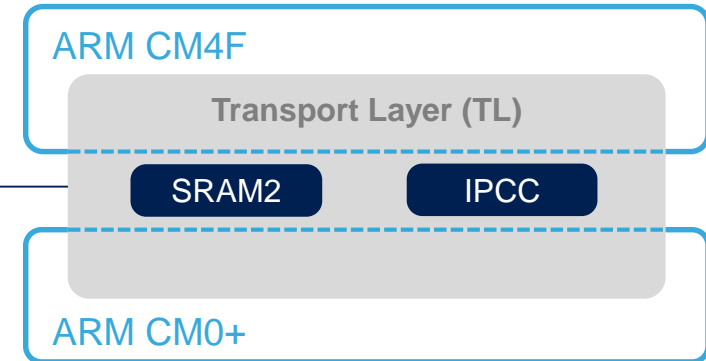
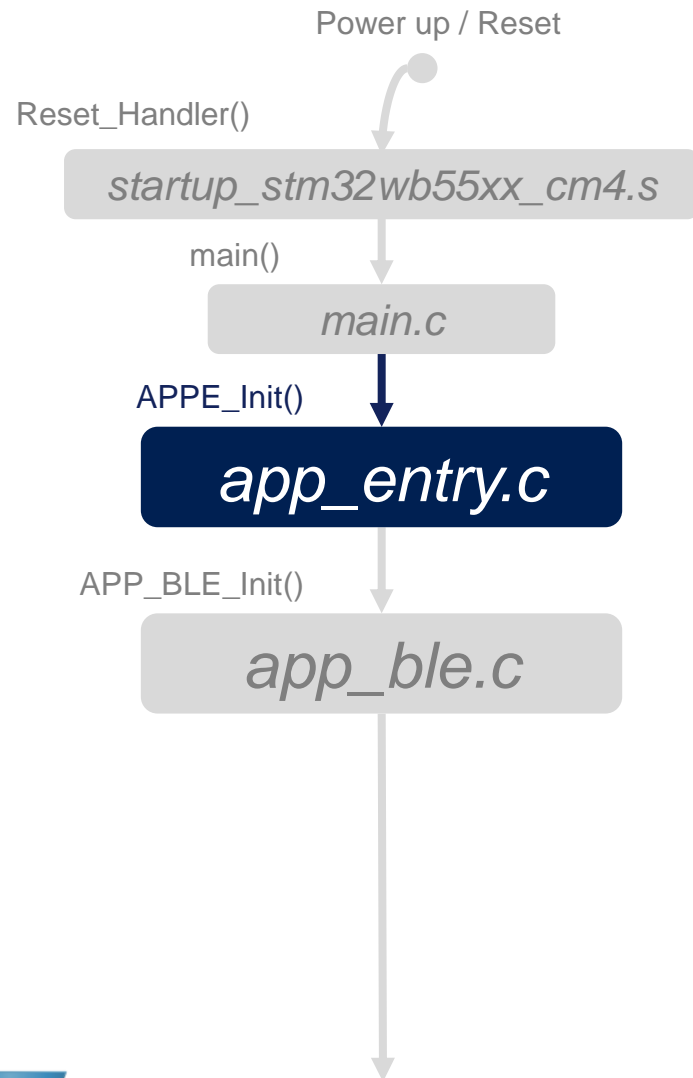
stm32wbxx_hal_gpio.c

```
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);
  }
}
```

TO BE IMPLEMENTED IN END USER CODE



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;
}
```




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

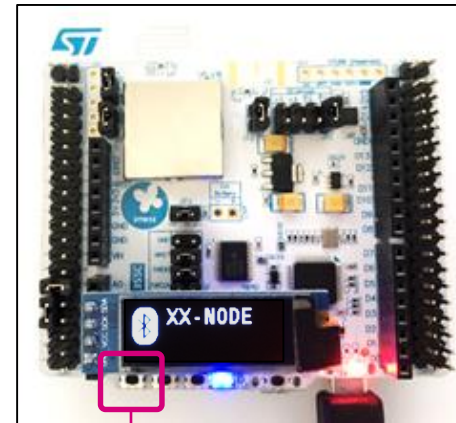
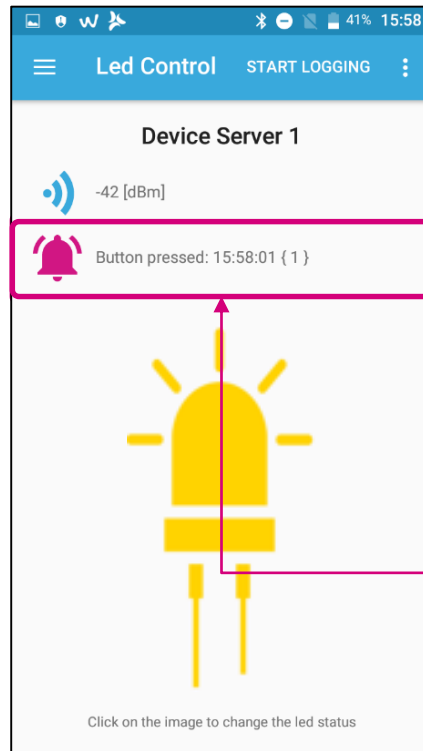
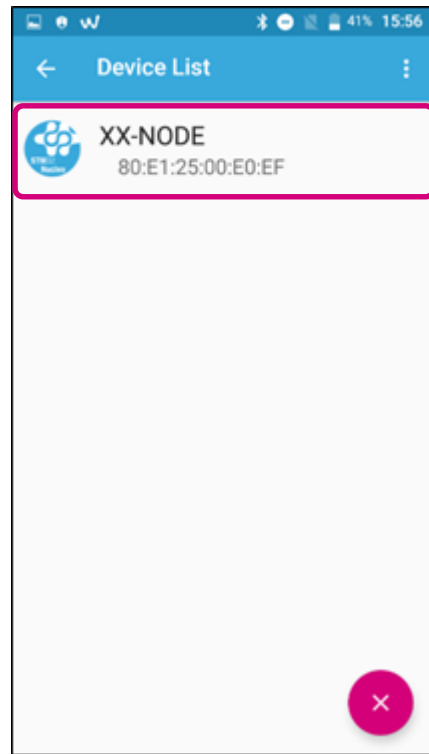
We are still in ISR !

Proper Non-Blocking implementation

```
/* USER CODE END FD */
```

Test the functionality

- 1 Build**

or Ctrl+B
- 2 Debug**

F11
- 3 Resume**

F8





Central GATT Client

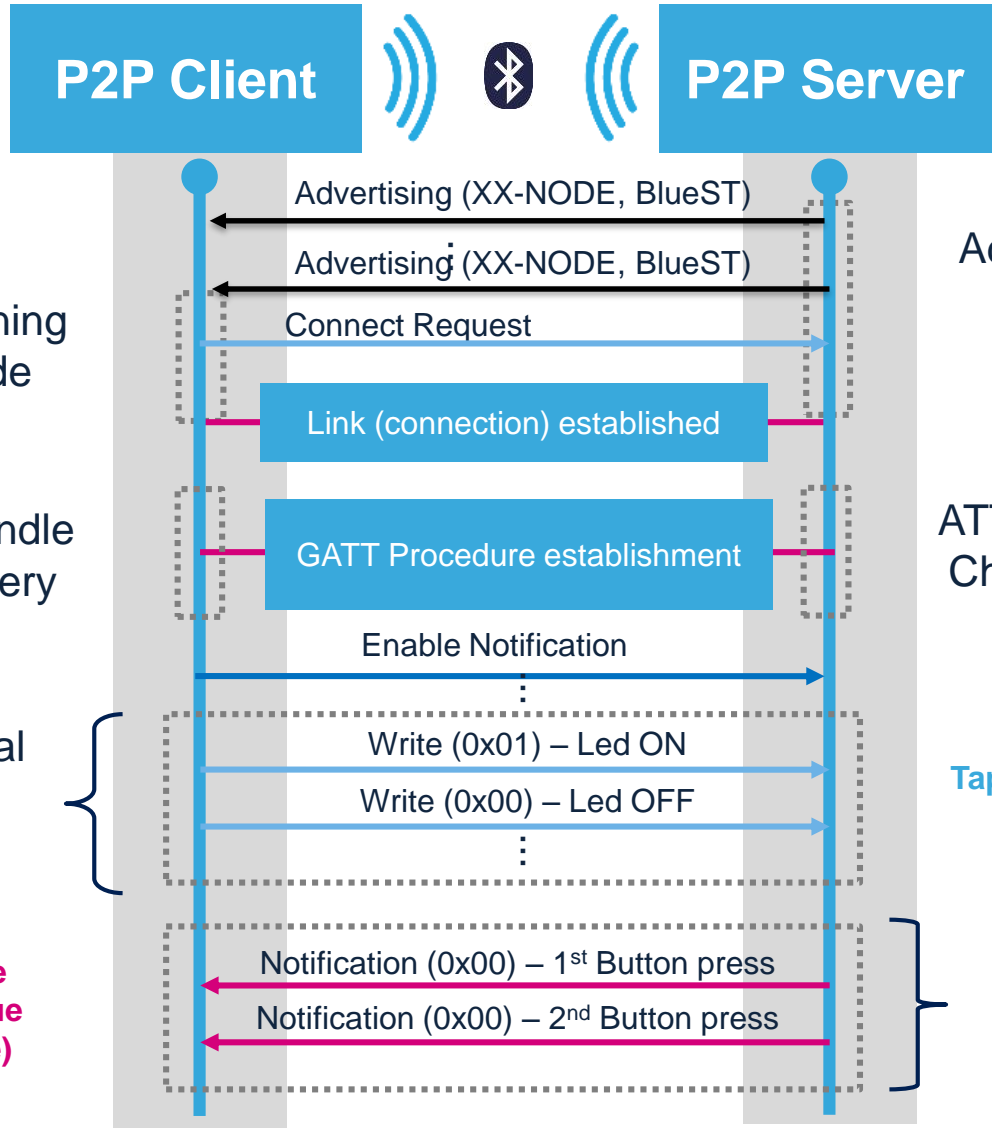


Central-to-Peripheral communication (Control SLAVE)

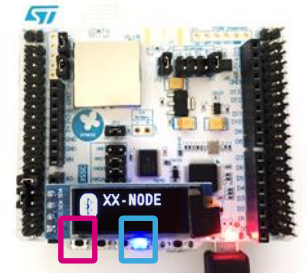
Push SW1 to update characteristic #1 value (notify value change)



Finished !



Peripheral GATT Server



Advertising Mode

ATT Services & Characteristics

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

Peripheral-to-Central communication (Notify MASTER)



GAP and GATT commands and events used so far

164

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(...);`

Attribute modified by client → `EVT_BLUE_GATT_ATTRIBUTE_MODIFIED`



모든 제품



대한민국

한국어

KRW

제품

제조업체

고객 지원

도구

로그인 또는
등록

0 개 품목

60,000원 또는 미화 50달러 이상 주문 시 무료 선적!

제품 색인 > RF/IF 및 RFID > RF 평가 및 개발 키트, 기판 > STMicroelectronics P-NUCLEO-WB55

즐거찾기에 추가 ☆ 공유

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

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 요청서](#)를 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	<input type="radio"/>
제조업체	STMicroelectronics	<input type="checkbox"/>
계열	-	<input type="checkbox"/>



Releasing Your Creativity



 /STM32

 @ST_World

 ST community



www.st.com/stm32wb