



# ST67W611 Wi-Fi 6 + Bluetooth® LE + Thread

Santa Clara Wireless Team

# Agenda

# X-Cube-ST67W61 Package

# Device Setup and Flashing

# BLE Commissioning App

# Generating BLE Commissioning app using NUCLEO-U575ZI-Q as host processor (Hands-on)

# Porting to a new host mcu using STM32CubeMX

# Generating the BLE Commissioning app using NUCLEO-G0B1RE processor (Hands-on)

# Documentation

# Prerequisites

**PLEASE DOWNLOAD ALL THE TOOLS LISTED BELOW**

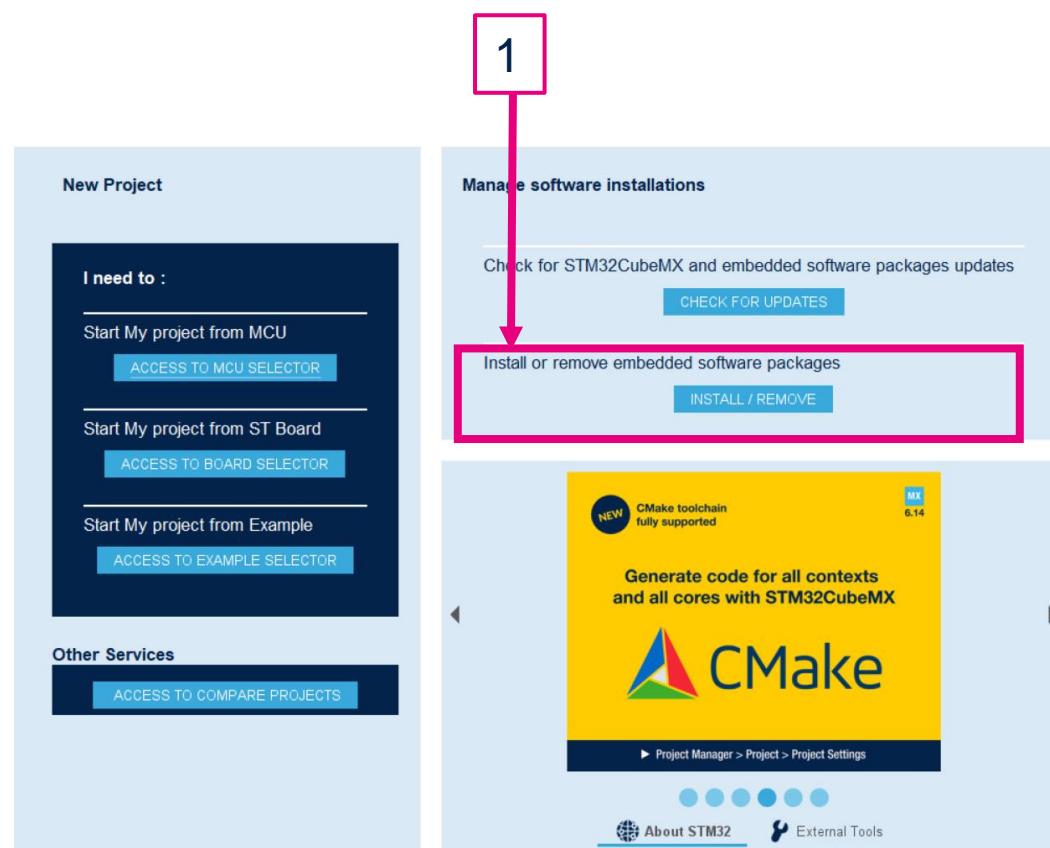
- [X\\_CUBE\\_ST67W61\\_V1.0.0](#) (To be installed under C drive)
- [STM32CubelDE v1.16.1+](#)
- [STM32CubeProgrammer v2.18.0+](#) To be installed in default directory  
(C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer)
- [STM32CubeMX v6.15.0](#)
- [HyperTerminal – TeraTerm v.5.4.0](#)
- [ST BLE Toolbox](#) (To be downloaded on mobile)



# Installing X-CUBE-ST67W61

## Open STM32CubeMX with admin rights

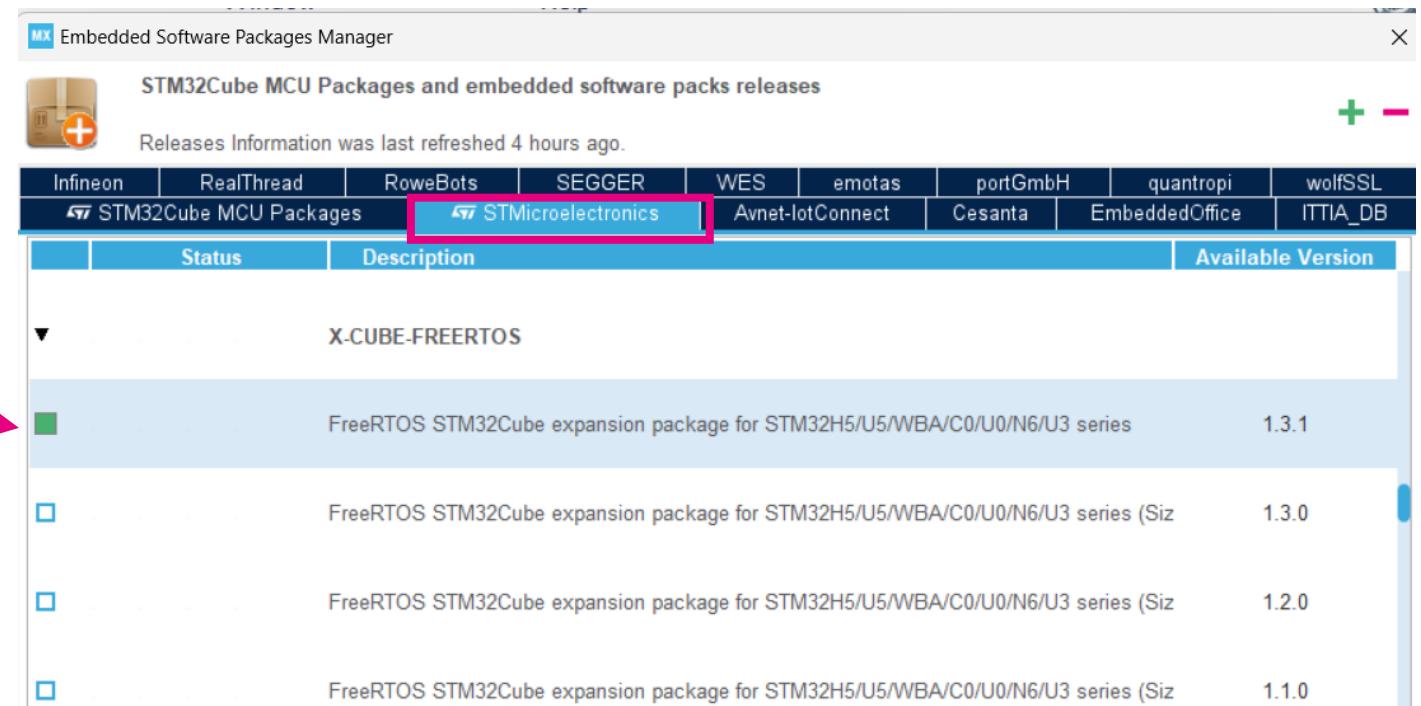
1. Go to “Install/Remove” software packages
2. Under STMicroelectronics tab
3. Install the X-CUBE-ST67W61



The image shows the 'Embedded Software Packages Manager' window. At the top, it says 'STM32Cube MCU Packages and embedded software packs releases'. A red box highlights the 'STM32Cube MCU Packages' tab, with a red number '2' above it. Below this is a table with columns for 'Status', 'Description', and 'Available Version'. The 'Description' column lists several packages: X-CUBE-SMBUS, X-CUBE-ST60, X-CUBE-ST67W61 (which is highlighted with a red box and a red number '3' above it), and X-CUBE-SUBG2. The 'Available Version' column shows 1.0.0 for X-CUBE-ST67W61. At the bottom, there's a 'Details' section with the same text as the table.

# X-CUBE-FREERTOS

[Download latest X-CUBE-FREERTOS](#)



# STM32U5

MX Embedded Software Packages Manager

STM3Cube MCU Packages and embedded software packs releases

Releases Information was last refreshed 4 hours ago.

Infineon	RealThread	RoweBots	SEGGER	WES	emotas	portGmbH	quantropi	wolfSSL
STM3Cube MCU Packages	STMicroelectronics	Avnet-IoTConnect	Cesanta	EmbeddedOffice	ITIA_DB			
Description	Installed Version			Available Version				
▼ STM32U5								
<input checked="" type="checkbox"/> STM3Cube MCU Package for STM32U5 Series	1.8.0			1.8.0			<a href="#">Download latest STM32U5</a>	
<input type="checkbox"/> STM3Cube MCU Package for STM32U5 Series (Size : 390 MB)				1.7.0				
<input type="checkbox"/> STM3Cube MCU Package for STM32U5 Series (Size : 351 MB)				1.6.0				
<input type="checkbox"/> STM3Cube MCU Package for STM32U5 Series (Size : 306 MB)				1.5.0				

# STM32G0

MX Embedded Software Packages Manager

STM32Cube MCU Packages and embedded software packs releases

Releases Information was last refreshed 20320 days ago.

Infineon	RealThread	RoweBots	SEGGER	WES	emotas	portGmbH	quantropi	wolfSSL
STM32Cube MCU Packages		STMicroelectronics		Avnet-IoTConnect		Cesanta	EmbeddedOffice	ITIADB
<b>Description</b>								
▶ STM32F4								
▶ STM32F7								
▼ STM32G0								
■ STM32Cube MCU Package for STM32G0 Series						1.6.2	1.6.2	
□ STM32Cube MCU Package for STM32G0 Series (Size : 204.68 MB)							1.6.1	
□ STM32Cube MCU Package for STM32G0 Series (Size : 203 MB)							1.6.0	

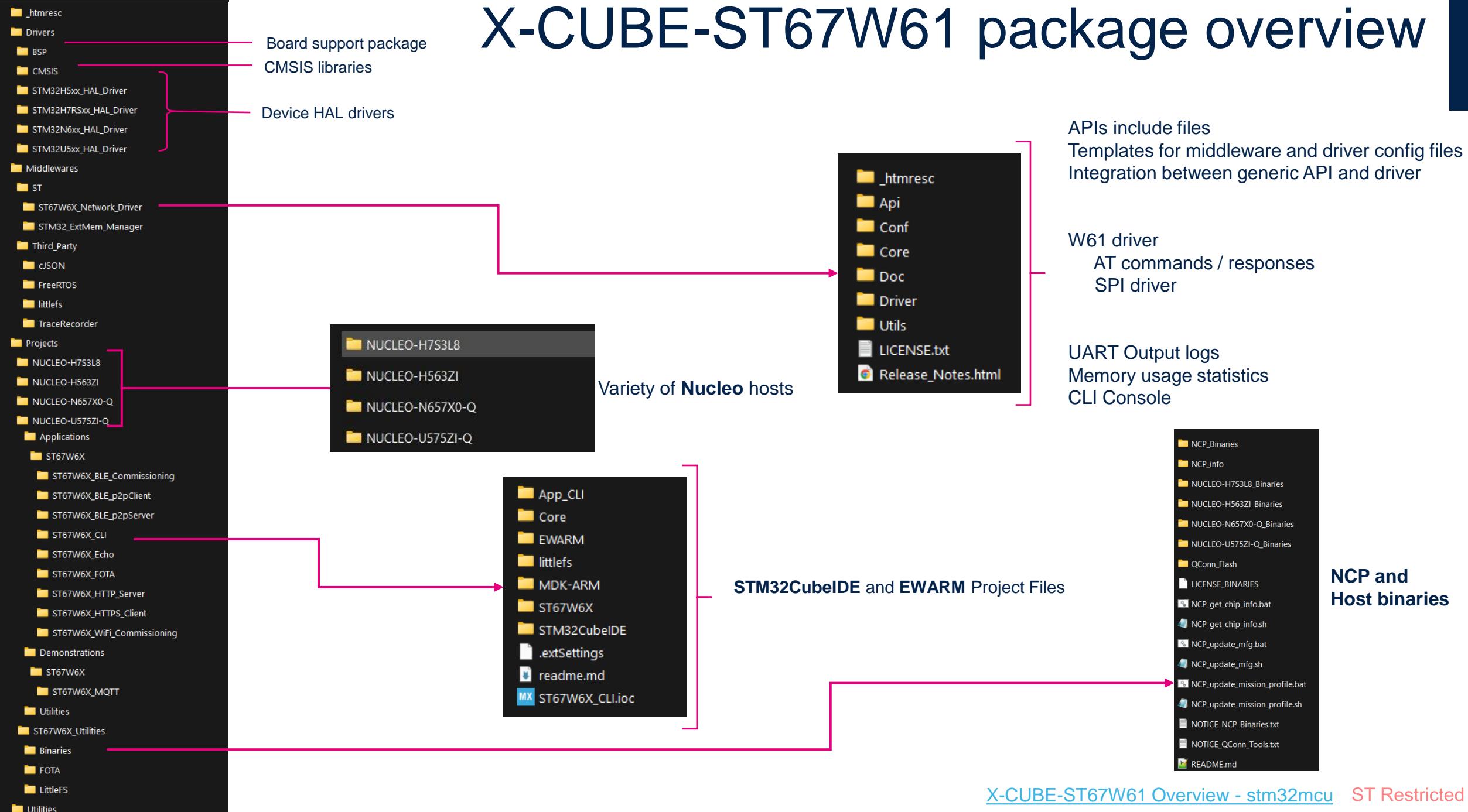
Download latest STM32G0





# X-Cube-ST67W61

# X-CUBE-ST67W61 package overview

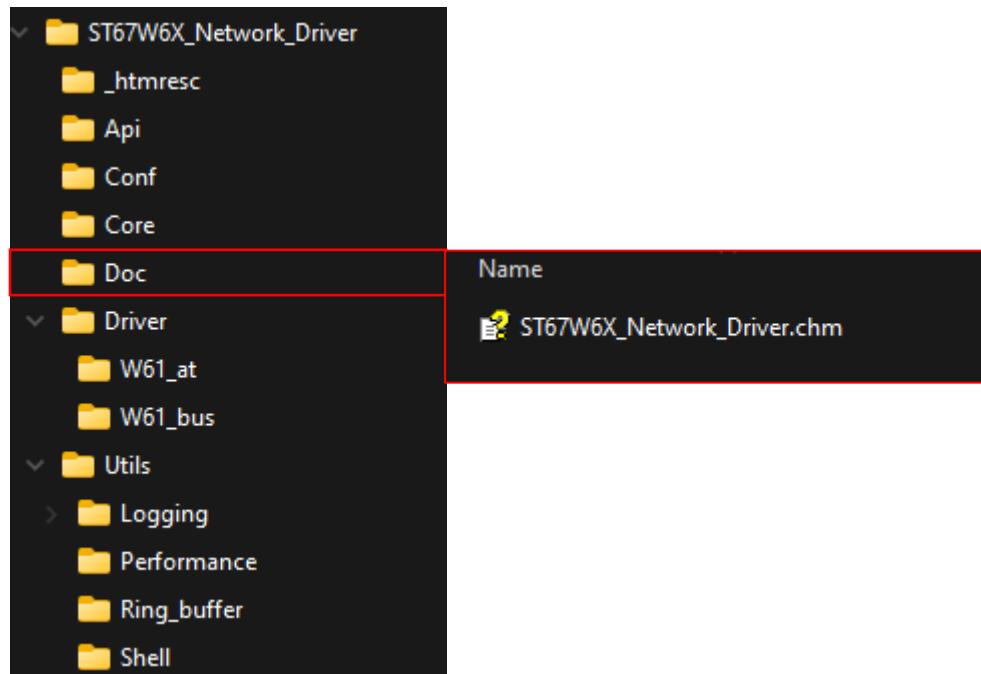
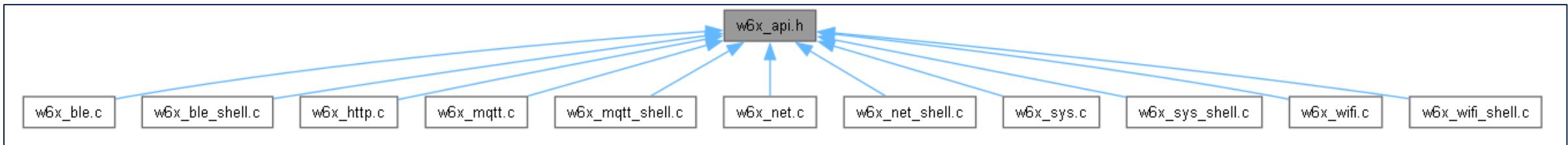


# X-CUBE-ST67W61 middleware

	Scope / description
W61_bus	This is the <b>SPI driver for communication</b> between the host and the ST67W61.
W61_at	It does the <b>formatting of the AT commands</b> to be sent, and it implements a receive tasks <b>to decode the received messages</b> from the W61 and dispatch them to the correspondent module.
W6x service API	The Core directory implements an adaptation layer between the W61 AT driver and the API proposed to the application. The APIs are "Zephyr-like" for the Wi-Fi® and socket API for the network operations.
Utils	Console handling, debug, trace, performance measurements.

All the above submodules make use of FreeRTOS™

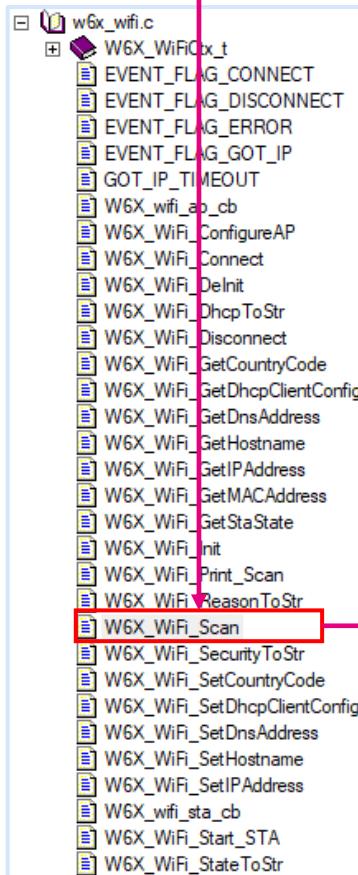
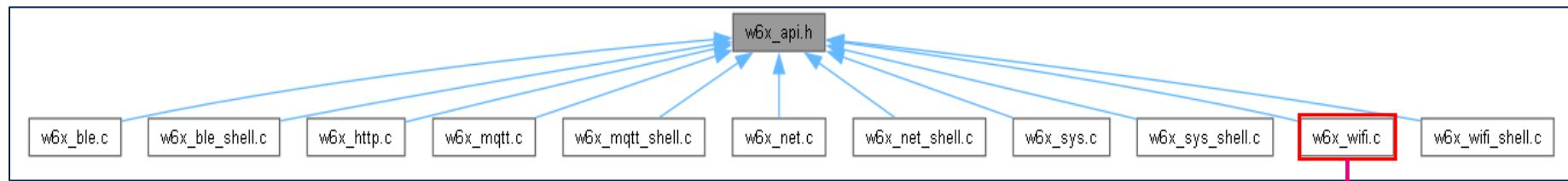
# API Documentation



Found in \Middlewares\ST\ST67W6X\_Network\_Driver\Doc

- API Definitions
- Overview of API calls
- Correlated AT Commands

# API Documentation cont...



Selecting **w6x\_wifi.c** will provide the different API categories

Selecting the API, **W6X\_WiFi\_Scan()**, will provide information and the call graph

◆ **W6X\_WiFi\_Scan()**

```
W6X_Status_t W6X_WiFi_Scan (W6X_Scan_Opts_t * Opt,
                             scan_result_cb_t cb)
```

List a defined number of available access points.

**Parameters**

- Opts Scan options
- cb Callback to handle scan results

**Returns**

- Operation status

Definition at line 260 of file [w6x\\_wifi.c](#).

References [W61\\_WiFi\\_Scan\(\)](#), and [W61\\_WiFi\\_SetScanOpts\(\)](#).

Referenced by [W6X\\_Shell\\_WiFi\\_Scan\(\)](#).

Here is the call graph for this function:

```
graph TD; W6X_WiFi_Scan[W6X_WiFi_Scan] --> W61_WiFi_Scan[W61_WiFi_Scan]; W6X_WiFi_Scan --> W61_WiFi_SetScanOpts[W61_WiFi_SetScanOpts]; W61_WiFi_Scan --> AT_SetExecute[AT_SetExecute]; W61_WiFi_SetScanOpts --> AT_SetExecute; AT_SetExecute --> AT_ParseOkErr[AT_ParseOkErr]; AT_SetExecute --> ATrrev[ATrrev]; AT_SetExecute --> ATLock[ATLock]; AT_SetExecute --> ATSend[ATsend]; ATLock --> ATUnlock[ATUnlock]; ATUnlock --> ATLock;
```

# X-CUBE-ST67W61 applications

Applications
ST67W6X
ST67W6X_BLE_Commissioning
ST67W6X_BLE_p2pClient
ST67W6X_BLE_p2pServer
ST67W6X_CLI
ST67W6X_Echo
ST67W6X_FOTA
ST67W6X_HTTP_Server
ST67W6X_HTTPS_Client
ST67W6X_WiFi_Commissioning
Demonstrations
ST67W6X
ST67W6X_MQTT

	Scope / description
Bluetooth® LE commissioning	Uses Bluetooth® LE to provide Wi-Fi® SSID and password after scanning available AP
Bluetooth® LE p2p client	Point-to-Point communication demo using Bluetooth® LE (GATT Client) Scan and connect to a p2pServer Write messages & receive notifications from the p2pServer
Bluetooth® LE p2p server	Point-to-Point communication demo using Bluetooth® LE (GATT server) Advertises and wait for a connection from either: - p2pClient app - ST BLE Toolbox smartphone application
Echo	TCP Echo feature over Wi-Fi - Good starting point for generic user development
CLI	ST67 evaluation via CLI and for WTS tests the Wi-Fi solution. - Wi-Fi API: scan, connect, disconnect, etc - NET socket API and iperf - Bluetooth® LE API
Echo FOTA	FOTA (firmware update over-the-air) feature demo over Wi-Fi - Using HTTP
HTTP server	HTTP server over Network API The server, running on the host board, has been validated with 1 client, using HTTP requests to send and receive data to and from the server.
HTTPS client	HTTPS Client over Network API demo - Basic GET requests to retrieve worldwide weather reports
Wi-Fi® commissioning	Wi-Fi® commissioning project using an HTTP server, hosted by a device configured as a soft access point (SoftAP) and station (STA).
MQTT Demo	Starting point for MQTT user development - Sends IKS sensor data to an MQTT broker



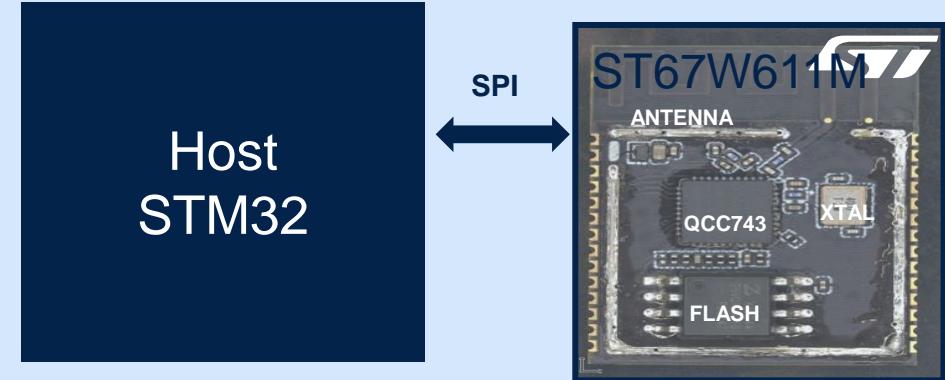
# ST67W611M Modes of Operation

## Manufacturing and Mission Modes

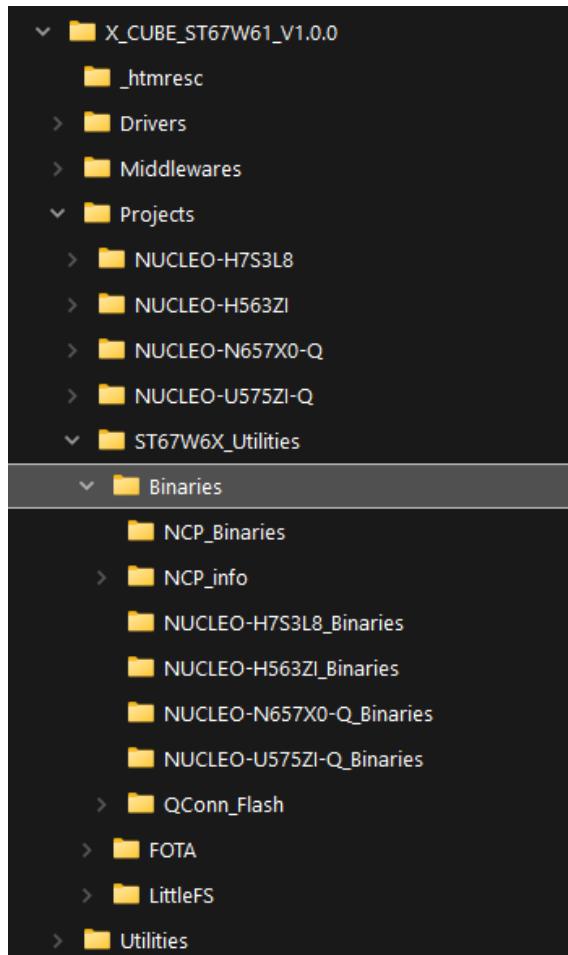
- **Manufacturing mode**
  - Host-less Application
  - Dedicated to RF performances monitoring and calibration.
  - QC provides graphical tools to control the device through the **UART** interface



- **Mission mode (Functional)**
  - Implementing the AT command for the different supported protocol (WIFI / BLE / MQTT /..) through the **SPI** interface.
  - The ROM boot loader loads and executes the program from SPI Flash to boot the system.



# X-CUBE-ST67W1 NCP tools



Scripts for flashing the latest mission or manufacturing firmware:

**..\\x-cube-st67w61-v1.0.0\\Projects\\ST67W6X\_Utilsities\\Binaries**

Tool	Description
<b>NCP_get_chip_info.bat</b>	Script to retrieve ST67 MAC address and locking information even if no firmware is loaded onto the ST67 device.
<b>NCP_update_mfg.bat</b>	Script to flash the ST67 module with the manufacturing application, which is used for RF evaluation.
<b>NCP_update_mission_profile.bat</b>	Script to flash the ST67 module with the mission profile application, which is used for normal operations and development purposes.

# ST67W611M Security Overview

**Two modes, each with a header containing a digital signature and versioning info**

## A. Mission mode:

- Bootloader binary
  - Authenticated by the ROM code during initial boot process
  - Versioning info to prevent rollback
- Mission profile binary
  - Wi-Fi® & Bluetooth® LE binary used for operational tasks
  - Authenticated by bootloader to ensure integrity & authenticity before execution

## B. Manufacturing mode:

- Bootloader binary
  - Authenticated by the ROM code during initial boot process
  - Versioning info to prevent rollback
- MFG firmware binary
  - Used for test and production configuration
  - Authenticated by bootloader to ensure integrity & authenticity before execution

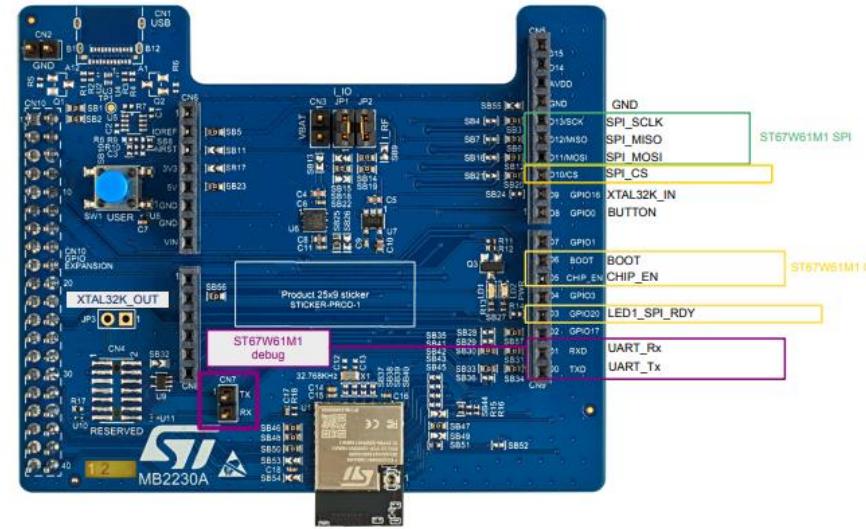




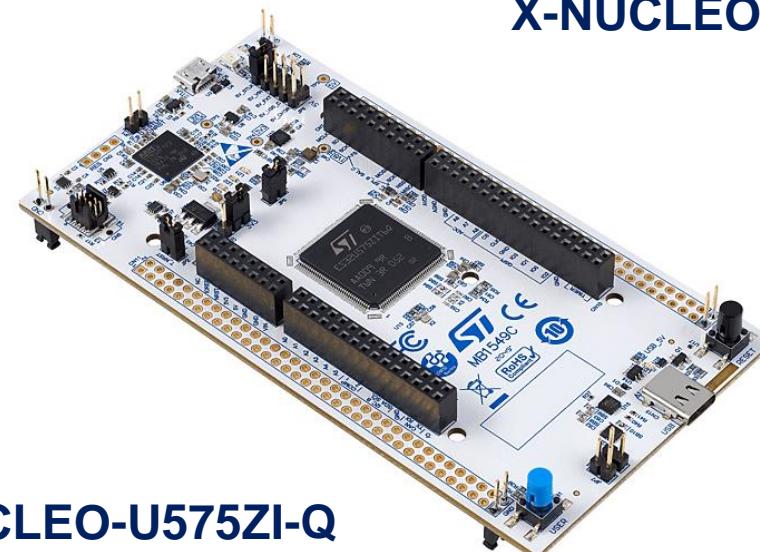
# Flashing ST67 & Device Setup

# STM32U575 and ST67W611M configuration

- 1x NUCLEO-U575ZI-Q board
- 1x X-NUCLEO-67W61M1 board
- Boards connected through Arduino® connectors
  - Key signals: SPI (SCK, MISO,MOSI,CS), SPI\_RDY, CHIP\_EN, BOOT, UART\_TX/RX
- Power: MicroUSB connection to NUCLEO-U575ZI-Q



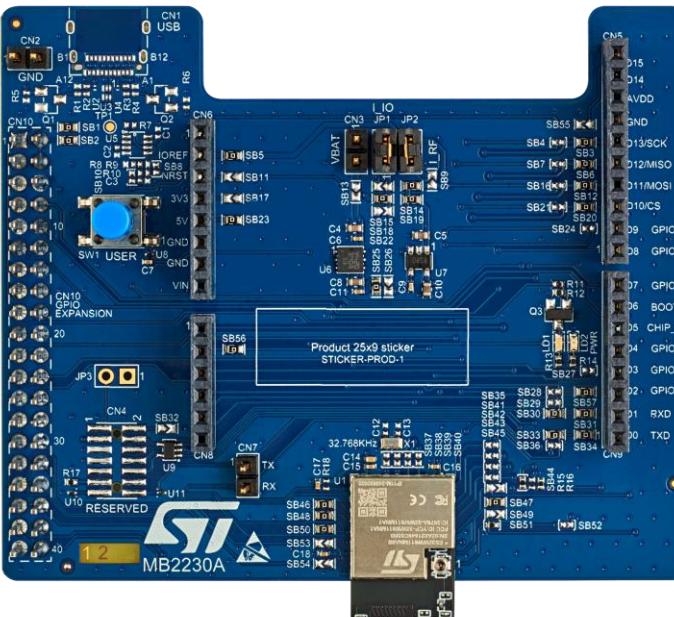
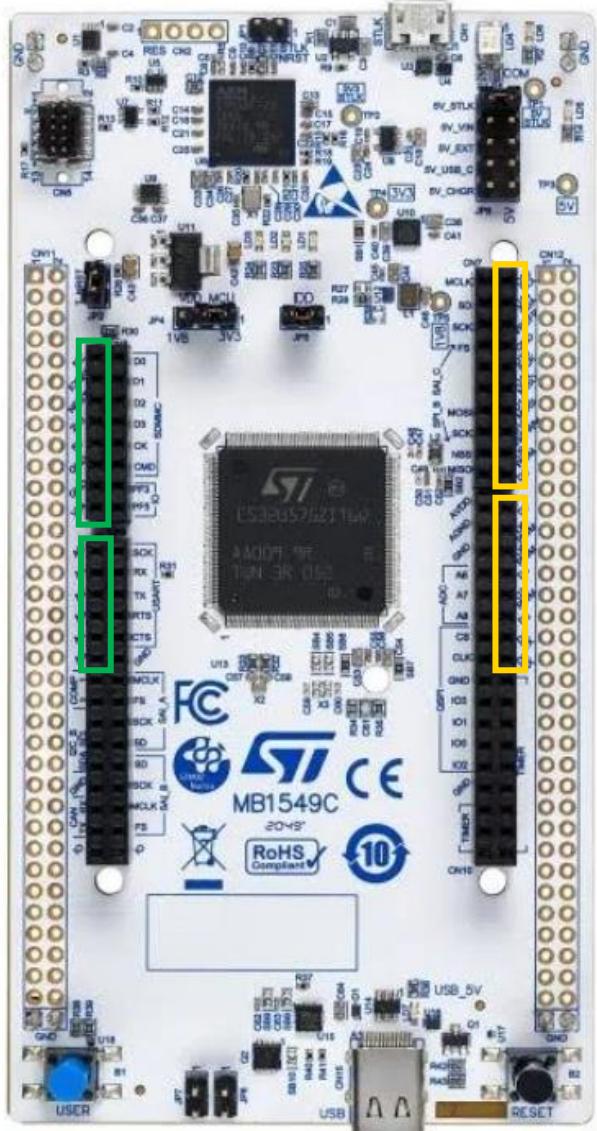
X-NUCLEO-67W61M1



NUCLEO-U575ZI-Q



# Nucleo-U575ZI and X-Nucleo-67W61M1 Configuration



When stacking the Nucleo and X-Nucleo boards the Arduino connectors must be aligned.

The MicroUSB port is used when programming the STM32U575 or ST67W611M1.

The MicroUSB port is also used to read the UART messages from the STM32U575 in a serial terminal.

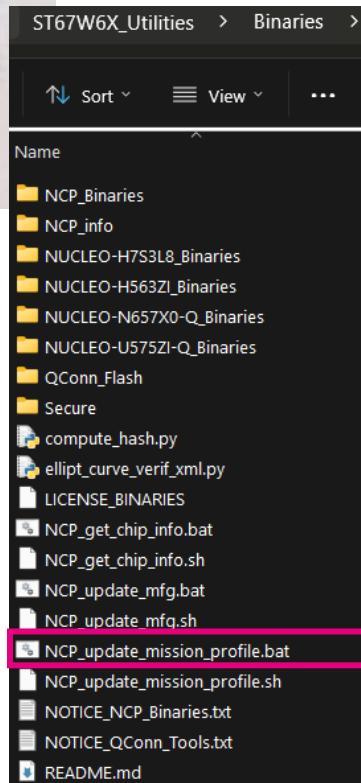
# Flashing ST67W611M1

## Mission Mode (Functional Mode) Setup



### Prerequisites:

- Installed STM32CubeProgrammer
  - Installed in default directory (C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer)
- Installed X-CUBE-ST67W61
  - Installed on C: drive (C:\x-cube-st67w61-v1.0.0)\*\*



### Using batch scripts located:

C:\x-cube-st67w61-v1.0.0\Projects\ST67W6X\_Utilsities\Binaries

1. Nucleo-U575ZI-Q and X-Nucleo-67W61M1 stacked together
2. USB cable connected to microUSB port on Nucleo-U575ZI-Q and Windows PC.
3. Double-click the batch files to execute
  1. NCP\_update\_mission\_profile.bat will flash the ST67W61M with the spi\_wifi binary and the host processor with CLI application.



\*\*Note: Path length limits and paths with spaces in them for the x-cube-st67w61-v1.0.0 may cause issues. Saving on C: drive will prevent these issues.

ST Restricted

```
"#####
## You are about to load a signed binary to the NCP."
## This will lock the ST67W61M if not yet locked."
#####
Are you sure to proceed? (Y/N)
Press Y to continue or N to abort: |
```

## Flashing ST67W611M1 Mission Mode (Functional Mode) Process

1. When executing NCP\_update\_mission\_profile.bat the script will prompt the user with a warning message about locking the NCP. This is expected, and all devices have previously been locked with older firmware at manufacturing. Please press “Y” to continue.

```
Opening and parsing file: Bootloader.bin

Memory Programming ...
File      : Bootloader.bin
Size      : 8.96 KB
Address   : 0x08000000

Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 1]
Download in Progress:

100%

File download complete
Time elapsed during download operation: 00:00:00.168
```

```
[11:54:50.657] - Load efuse 0
[11:54:50.657] - Load efuse 1
[11:54:50.657] - Load efuse remainder
[11:54:50.657] - Finished
[11:54:50.657] - All time cost(ms): 16587.53369140625
[11:54:50.767] - close interface
[11:54:50.767] - [All Success]
```

```
Opening and parsing file: ST67W6X_CLI.bin

Memory Programming ...
File      : ST67W6X_CLI.bin
Size      : 186.91 KB
Address   : 0x08000000

Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 23]
Download in Progress:

100%

File download complete
Time elapsed during download operation: 00:00:02.186
```

2. The NCP\_update\_mission\_profile.bat will have 3 sequential flashing stages. The script will flash the STM32U5 with Bootloader.bin, then the ST67 with the signed mission mode binary, and the STM32U5 with ST67W6X\_CLI.bin. Please verify success of all 3 flashing stages.

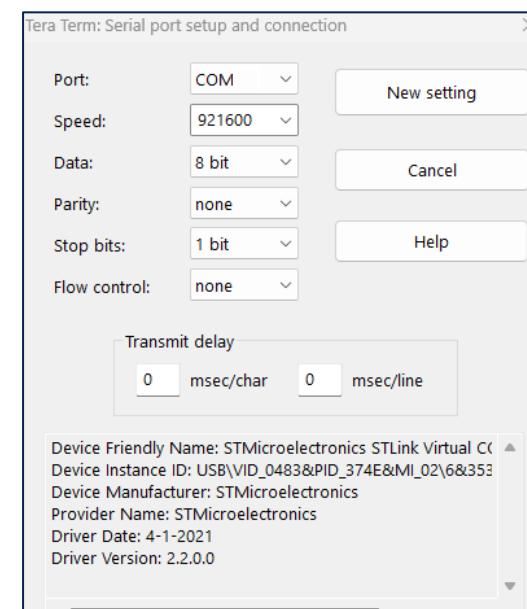
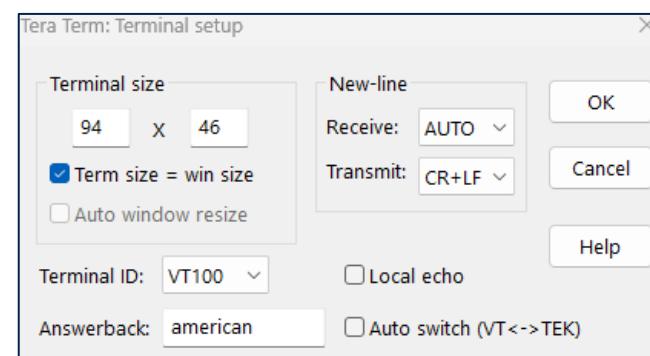


# Flashing ST67W611M1

## Mission Mode (Functional Mode) Verification

Flashing success can be verified by connecting to the Nucleo-U575ZI-Q with a serial terminal.

Configure the serial terminal with the following parameters and connect to the corresponding COM port for the Nucleo-U575ZI-Q.



With the serial terminal program configured, press the reset button the Nucleo-U575ZI-Q and verify application and SDK version.

```
>#### Welcome to ST67W6X CLI Application #####
# build: 22:39:24 May 28 2025
----- Host info -----
Host FW Version: 1.0.0
----- ST67W6X info -----
ST67W6X MW Version: 1.0.0
AT Version: 1.0.0.1
SDK Version: 2.0.75
MAC Version: 1.6.38
Build Date: May 20 2025 23:01:38
Module ID: C6AFDBD111400004
BOM ID: 1
Manufacturing Year: 2024
Manufacturing Week: 47
Battery Voltage: 3.334 V
Trim Wi-Fi hp: 6,6,6,6,6,5,5,6,6,7,7,7
Trim Wi-Fi lp: 7,7,8,8,8,9,9,9,10,10,10,11,11,11
Trim BLE: 5,4,5,6,7
Trim XTAL: 37
MAC Address: 40:82:7b:00:33:26
Anti-rollback Bootloader: 0
Anti-rollback App: 0
-----
mount success
Wi-Fi init is done
Net init is done
MQTT init is done
Starting FOTA task
ready
Application started from bank 1
```



\*\*Note: "Manufacturing Year" date may be incorrect and populated with the year "2000". This will have no effect on the functionality. © 2024 STMicroelectronics



# How to use X-CUBE-ST67W61 within STM32CubeMX

# Software Development Tools



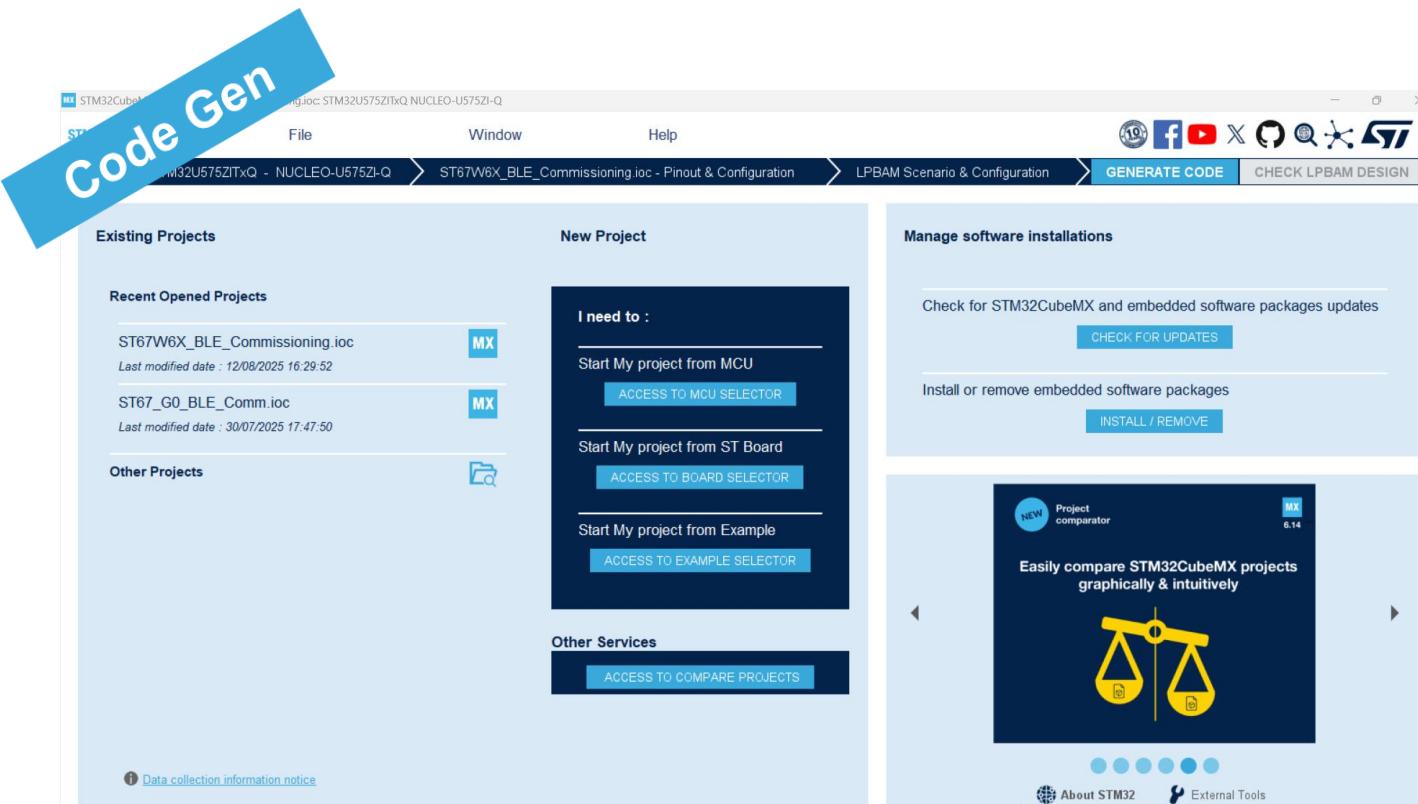
**“STM32CubeIDE”**  
development environment

↳ **“STM32CubeMX”**  
initialization code generator

↳ **“X-CUBE-ST67W61”**  
application code and drivers for Wi-Fi

# STM32CubeMX

an All-in-1 Development Tool



Very powerful configuration and code generation

Support all STM32 MCU and MPU, with integrated powerful Finder

Pinouts, clock tree, peripherals and middleware configuration.

Expandable to support wireless connectivity, sensors and much more!

Three possible cases to generate the code from the Cube MX:

- **Case 1:** How to generate a project starting from an existing .ioc
- **Case 2:** How to export a project to a pinout compatible MCU
- **Case 3:** How to generate a project starting from scratch

In this section: We would be having **hands-on** to generate **BLE Commissioning** code using the following:

- Case 1 with NUCLEO-U575ZI-Q as host processor
- Case 3 with NUCLEO-G0B1RE as host processor

# BLE Commissioning App

- This application aims to demonstrate **how to provision Wi-Fi credentials via Bluetooth® LE** to establish a Wi-Fi connection to an access point.



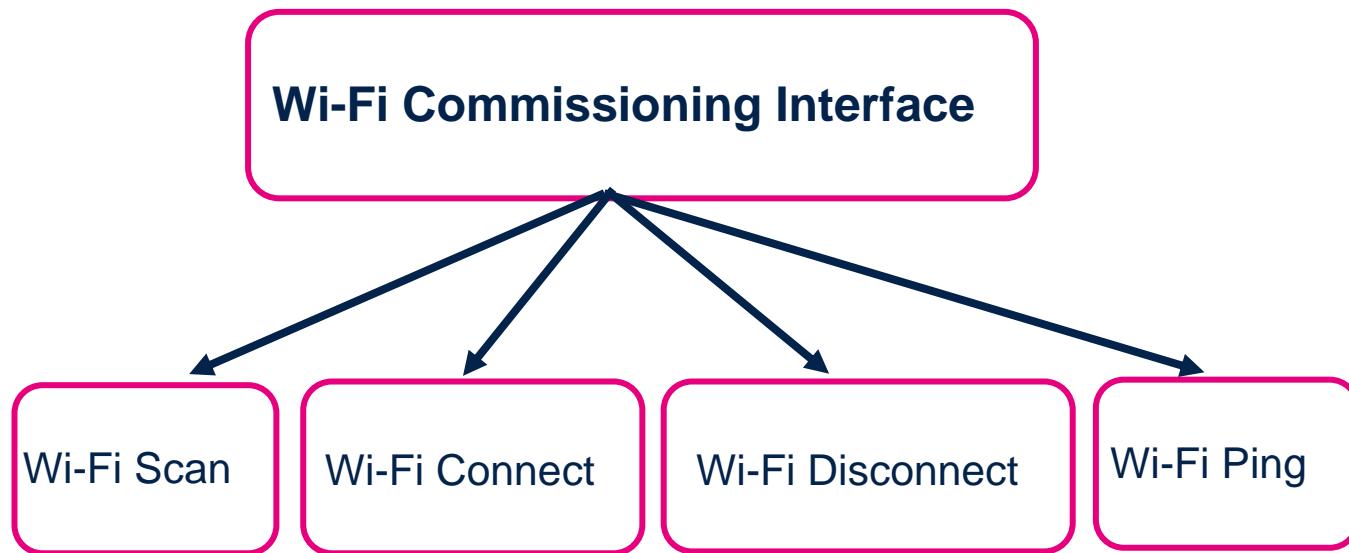
# BLE Commissioning App cont..

The application acts as a peripheral device embedding the commissioning profile and its four characteristics.

At startup, the application starts to advertise.

ST Web Bluetooth® Interface must be used to scan and connect to the commissioning application: (Web-Interface available from the browser) / ST BLE Toolbox (Mobile Application)

Once connected, a Wi-Fi commissioning interface is available on web Bluetooth® page/ ST BLE Toolbox



# Commissioning profile overview

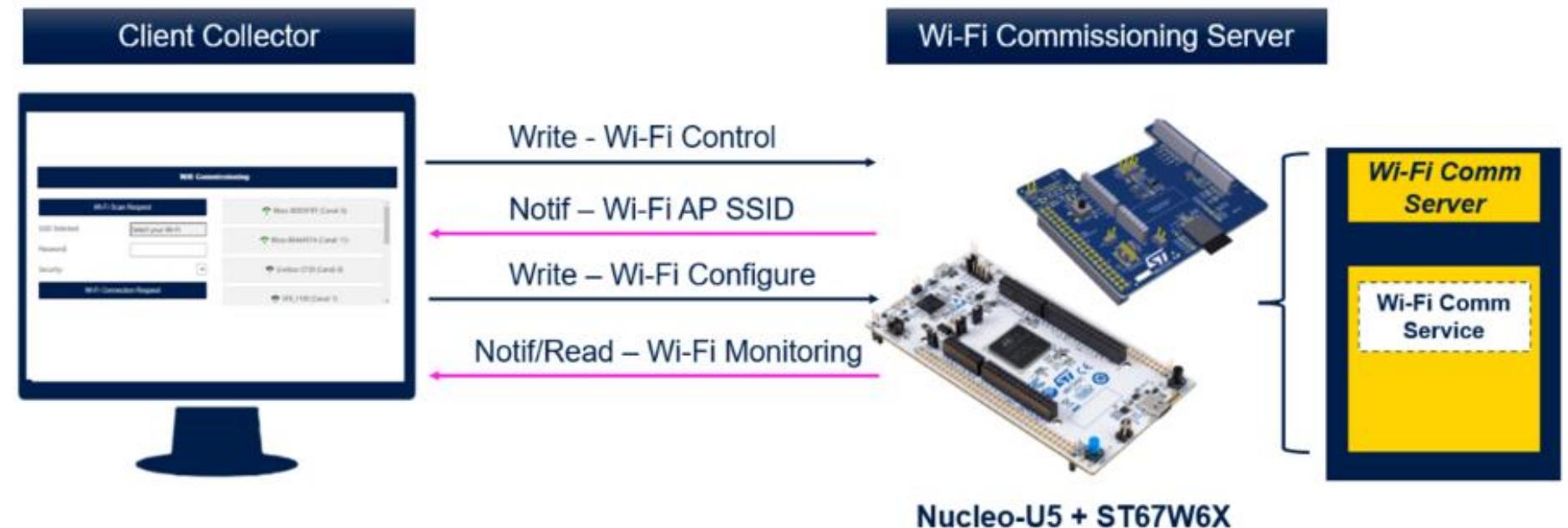
The Wi-Fi commissioning application demonstrates point to point communication using proprietary service and characteristics. It acts as a Peripheral device with one GATT service and four characteristics.

## Wi-Fi commissioning server:

- Contains the Wi-Fi commissioning service, which exposes four characteristics: **Wi-Fi Control (write)**, **Wi-Fi Configure (write)**, **Wi-Fi Access Point List (notification)**, **Monitoring (Notification)** to control and monitor a Wi-Fi connection.
- Is the GATT server

## Client Collector: (STBLE Toolbox/ ST Web Bluetooth page)

- Accesses the information exposed by the Wi-Fi commissioning application, **configures and controls the Wi-Fi connection** with the write characteristics, **monitors the Wi-Fi connection status** by receiving notifications from it
- Is the GATT client



The table below describes the structure of the commissioning service:

Bluetooth® LE commissioning service specification		
Service	Characteristic	Mode
Wi-Fi commissioning		
	Wi-Fi Control	Write with Response/Read
	Wi-Fi Configure	Write with Response/Read
	Wi-Fi AP List	Notify
	Monitoring	Notify

Wi-Fi commissioning - Wi-Fi control	
Byte Index	0
Name	Action
Value	0x01: Wi-Fi Start Scan 0x03: Wi-Fi Connect 0x04: Wi-Fi Disconnect 0x05: Wi-Fi Ping

**Wi-Fi control characteristic:**

Used to drive the Wi-Fi by launching scans, connecting or disconnecting from a network

### Wi-Fi commissioning - Wi-Fi configure

Byte Index	0	1
Name	Type	Data[ ]
Value	0x01: AP-SSID 0x02: PWD	"ASCII" "ASCII"

#### Wi-Fi configure characteristic:

Used to setup the Wi-Fi parameters before establishing a connection

### Wi-Fi Commissioning - Wi-Fi AP List

Byte Index	0	1	2 to 3	4 to 7	8 to 40
Name	SSID Length	Channel	Signal level	Security Flag	SSID
Value	0x00 ... 0x20	0x00 ... 0xFF	0x00 ... 0xFFFF	Security_Flag[ ]	Data[ ]

#### Wi-Fi access point list characteristic:

Used to notify information about scanned access points

### Wi-Fi commissioning - Wi-Fi monitoring

Byte Index	0	1 up to 239
Name	Type	Data[ ]
Value	0x03: Connecting 0x04: Connection established 0x05: Ping response 0x06: Error	X SSID Refer to table below 0x01: Connection Timeout

#### Monitoring characteristic:

Used to notify information about Wi-Fi network.

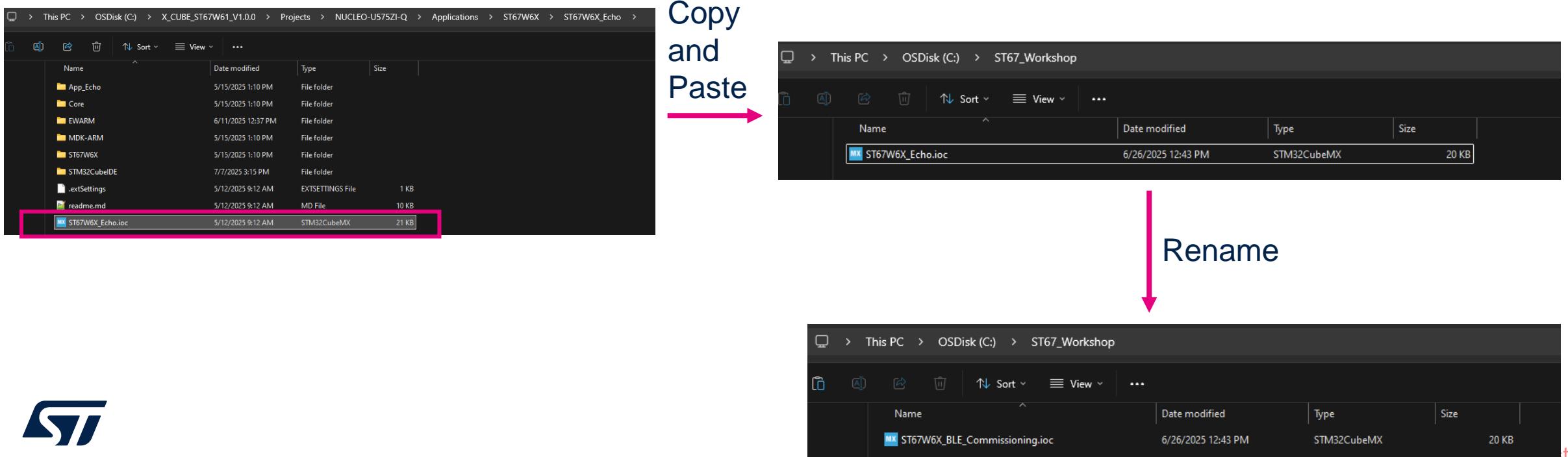


## Generating BLE Commissioning app using NUCLEO-U575ZI-Q as host processor (Hands-on)

# Case 1: Generating BLE Commissioning app using NUCLEO-U575ZI-Q as host processor

We would start with the ST67W6X\_Echo.ioc file as a starting point and change the configuration to generate the BLE Commissioning application from the STM32CubeMX.

1. Browse through C:\X\_CUBE\_ST67W61\_V1.0.0\Projects\NUCLEO-U575ZI\_Q\Applications\ST67W6X\ST67W6X\_Echo
2. Copy ST67W6X\_Echo.ioc
3. Create a new directory C:\ST67\_Workshop
4. Paste and rename to ST67W6X\_BLE\_Commissioning.ioc

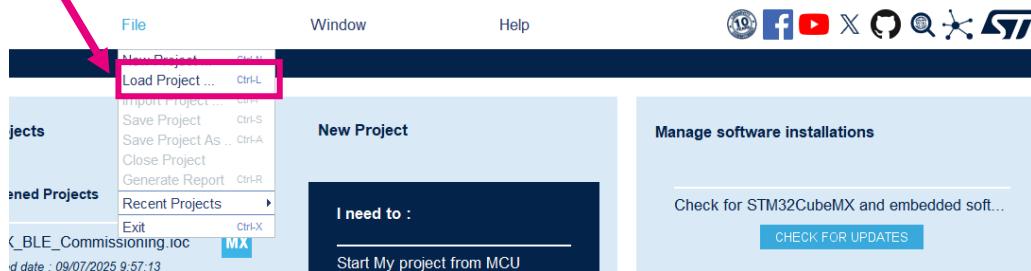


# Loading ioc file in STM32CubeMX

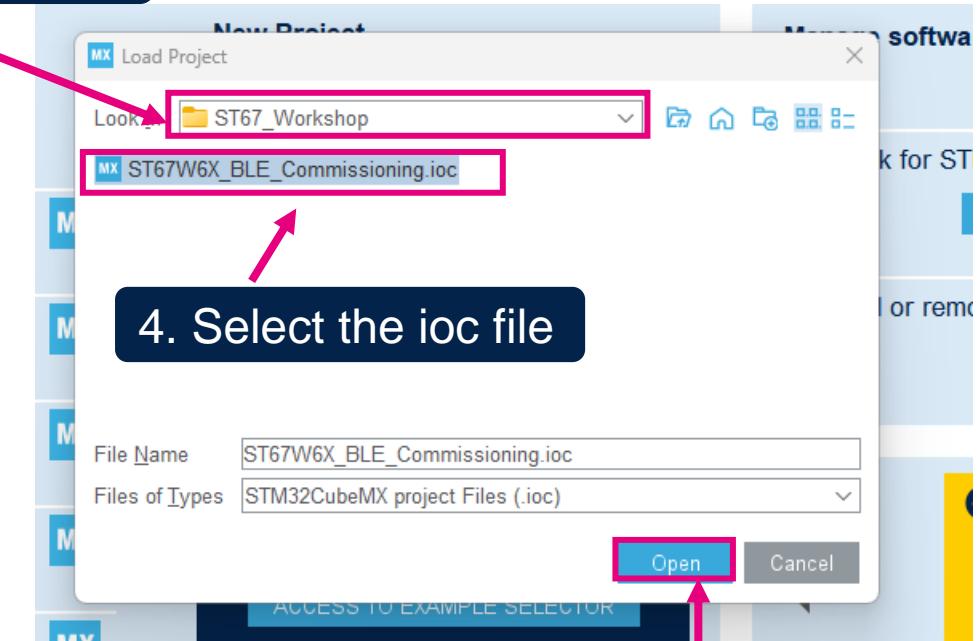
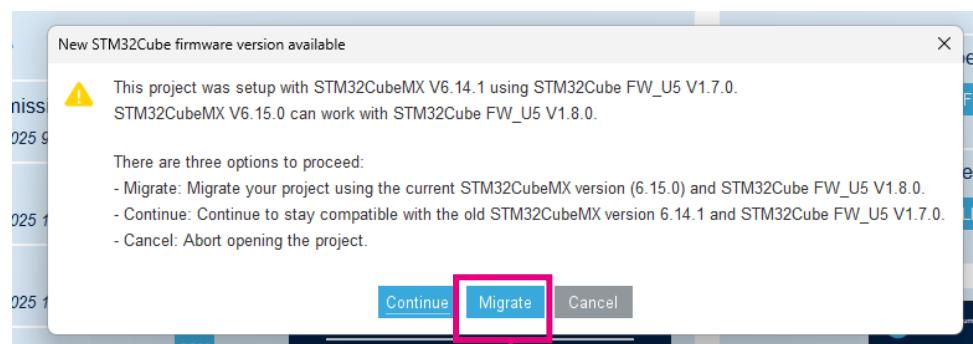
3. Go the ST67\_Workshop

1. Run STM32CubeMX with admin rights

2. Load Project

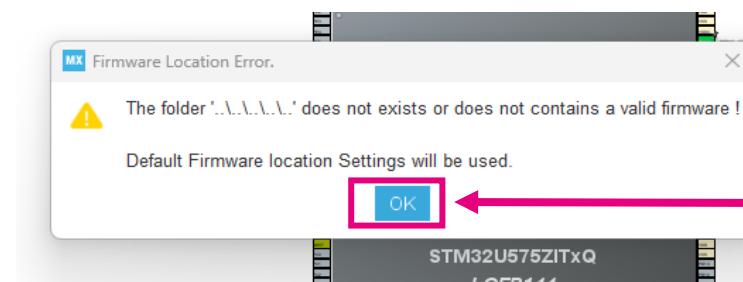


6. Select Migrate



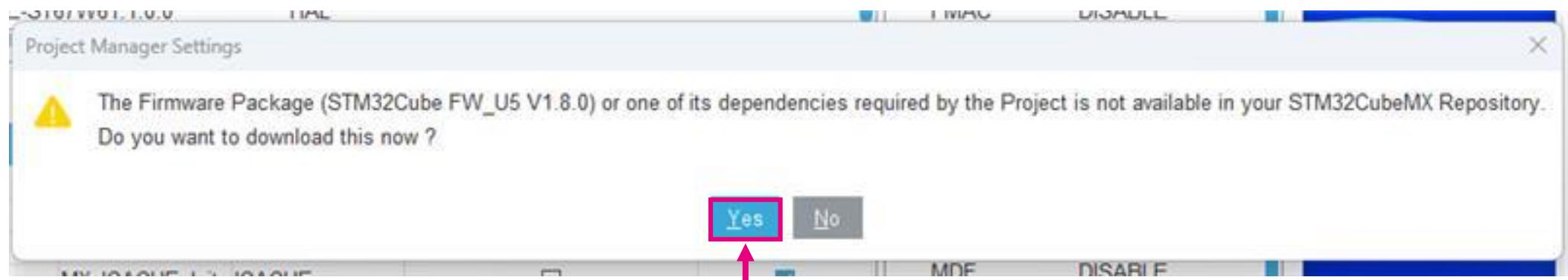
4. Select the ioc file

5. Click open



7. Click Ok

# If you get the below warnings

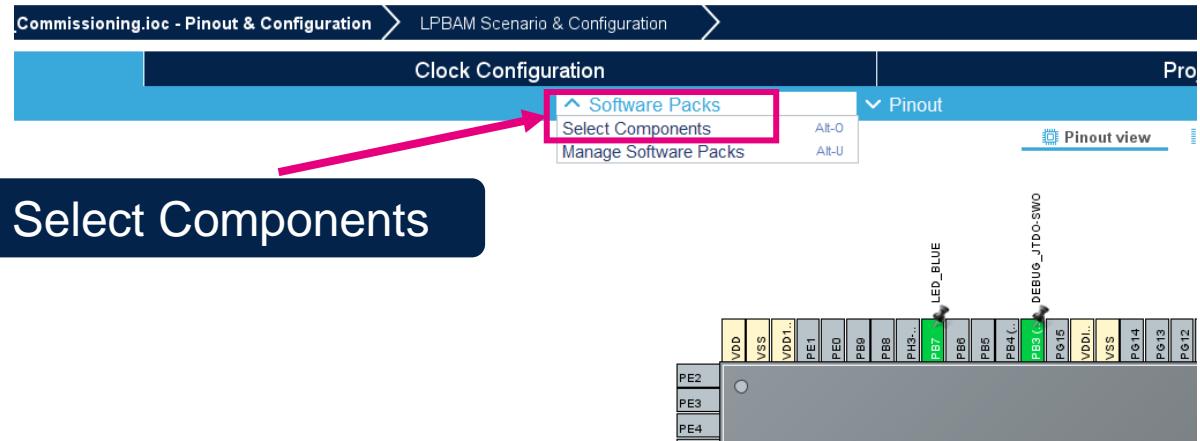


# Configuration

## 2. Select BLE Commissioning app

Packs	Status	Version	Selection
STMicroelectronics.X-CUBE-ST67W61	✓	1.0.0	
Device Applications	✓	1.0.0	
Application	✓	1.0.0	BLE_Commissioning_v
FreeRTOS_Tickless	✓	1.0.0	CLI_v
Network ST67W6X_Network_Driver	✓	1.0.0	BLE_Commissioning_v
ServiceShell		1.0.0	BLE_p2pClient_v
ServiceAPI	✓	1.0.0	BLE_p2pServer_v
Driver / W61_at_and_bus	✓	1.0.0	FOTA_v
Utilities	✓		HTTP_Server_v
Logging	✓	1.0.0	HTTPS_Client_v
Shell		1.0.0	MQTT_v
Statistics		1.0.0	
Debug TraceRecorder		4.10.2	
TraceRecorder		4.10.2	
Data Exchange cJSON		1.7.18	
cJSON		1.7.18	
File System LittleFS		2.10.1	
LittleFS		2.10.1	
Utility Utilities	✓	1.4.2	
LPM / Tiny LPM	✓	1.4.2	<input checked="" type="checkbox"/>
> STMicroelectronics.X-CUBE-SUBG2	5.0.0	↳	Install
> STMicroelectronics.X-CUBE-TCPP	4.2.0	↳	Install
> STMicroelectronics.X-CUBE-TOF1	3.4.3	↳	Install
> STMicroelectronics.X-CUBE-TOUCHGFX	4.25.0	↳	Install

## 1. Select Components



# Configuration

3. Under Middleware and Software Packs

Categories A->Z

Middleware and Software Packs

- AIROC-Wi-Fi-Bluetooth-STM32
- FILEX
- FP-SNS-FLIGHT1
- FP-SNS-MOTENV1
- FP-SNS-MOTENVB1
- FP-SNS-SMARTAG2
- FP-SNS-STAIOTCF1
- FP-SNS-STBOX1
- I-CUBE-CANOPEN
- I-CUBE-Cesium
- I-CUBE-FS-RTOS
- I-CUBE-ITTIADB
- I-CUBE-Mongoose
- I-CUBE-embOS
- I-CUBE-wolfMQTT
- I-CUBE-wolfSSH
- I-CUBE-wolfSSL
- I-CUBE-wolfTPM
- I-Cube-SoM-uGOAL
- LEVELX
- NETDUO
- THREADX
- TOUCHSENSING
- USBPD
- USBX
- X-CUBE-AI
- X-CUBE-ALGOBUILD
- X-CUBE-ALS
- X-CUBE-BLE1
- X-CUBE-BLE2
- X-CUBE-BLEMGR
- X-CUBE-DISPLAY
- X-CUBE-DPower
- X-CUBE-EEPRMA1
- X-CUBE-FREERTOS
- X-CUBE-GNSS1
- X-CUBE-IPS
- X-CUBE-ISPU
- X-CUBE-IoTC-DA16k-PMOD
- X-CUBE-MEMS1
- X-CUBE-NFC4
- X-CUBE-NFC6
- X-CUBE-NFC7
- X-CUBE-NFC9
- X-CUBE-NFC10
- X-CUBE-NFC12
- X-CUBE-PN33A1
- X-CUBE-RT-Thread\_Nano
- X-CUBE-SAFEA1
- X-CUBE-SFXS2LP1
- X-CUBE-SMBUS
- X-CUBE-ST100
- X-CUBE-ST67W61

4. X-CUBE-ST67W61

STMicroelectronics.X-CUBE-ST67W61.1.0.0 Mode and Configuration

Mode

Device Applications

Network ST67W6X Network Driver

Utility Utilities

Configuration

Reset Configuration

W6X Modules Parameter Settings User Constants Platform Settings

Configure the below parameters :

Basic Parameters

- DEBUGGER\_ON
- LOG\_OUTPUT\_MODE
- LOW\_POWER\_MODE
- FREERTOS\_TASK\_NOTIF\_ARRAY\_LEN

Logging Shell

- LOG\_LEVEL

W6X Parameters

- W6X\_BLE\_HOSTNAME
- W6X\_WIFI\_HOSTNAME
- W6X\_POWER\_SAVE\_AUTO
- W6X\_CLOCK\_MODE
- W6X\_WIFI\_AUTOCONNECT
- W6X\_WIFI\_DHCP\_MODE
- W6X\_WIFI\_COUNTRY\_CODE
- W6X\_WIFI\_ADAPTIVE\_COUNTRY\_CODE
- W6X\_NET\_RECV\_TIMEOUT (ms)
- W6X\_NET\_SEND\_TIMEOUT (ms)
- W6X\_NET\_RECV\_BUFFER\_SIZE
- W6X\_WIFI\_DNS\_MANUAL
- W6X\_WIFI\_DNS\_IP\_1
- W6X\_WIFI\_DNS\_IP\_2
- W6X\_WIFI\_DNS\_IP\_3

W61 dw Parameters

- W61\_WIFI\_MAX\_DETECTED\_AP
- W61\_BLE\_MAX\_CONN\_NBR
- W61\_BLE\_MAX\_DETECTED\_PERIPHERAL
- W61\_BLE\_MAX\_SERVICE\_NBR
- W61\_BLE\_MAX\_CHAR\_NBR

ON

LOG\_OUTPUT\_UART

LOW\_POWER\_SLEEP\_ENABLE

8

LOG\_DEBUG

TEST\_BLE

ST10/W61\_WiFi

No

Internal RC oscillator

No

DHCP client STA

00

No

10000

10000

9216

No

{208, 67, 222, 222}

{8, 8, 8, 8}

{0, 0, 0, 0}

50

1

10

5

5

5. Rename W6X\_BLE\_HOSTNAME to something unique

6. Disable W6X\_POWER\_SAVE\_AUTO

Bluetooth Low Energy connection with NCP in Power save mode requires accurate external low-frequency clock. To setup and use external clock, SW and HW settings must be modified.

Refer to [Wiki Bluetooth LE with low-power setup]([https://wiki.st.com/stm32mcu/wiki/Connectivity:How\\_to\\_measure\\_ST67W61M\\_current\\_consumption#Bluetooth-C2-AE\\_LE\\_with\\_low-power\\_setup](https://wiki.st.com/stm32mcu/wiki/Connectivity:How_to_measure_ST67W61M_current_consumption#Bluetooth-C2-AE_LE_with_low-power_setup)) page to be informed about required changes.

# Project Settings

Project Name: ST67W6X\_BLE\_Commissioning

Project Location: C:\ST67\_Workshop

Application Structure: Advanced

Toolchain Folder Location: C:\ST67\_Workshop\ST67W6X\_BLE\_Commissioning

Toolchain / IDE: STM32CubeIDE

Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Thread-safe Settings

CortexM33

Enable multi-threaded support

Thread-safe Locking Strategy: Default – Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

Mcu Reference: STM32U575ZITxQ

Firmware Package Name and Version: STM32Cube\_FW\_U5\_V1.8.0

Use Default Firmware Location

Firmware Relative Path: C:/Users/kulkarni/STM32Cube\_FW\_U5\_V1.8.0

1. Enable “Generate Under Root”

2. Enable “ Use Default Firmware Location”

Here, we are enabling the Flat directory structure: (i.e. Driver and MW directories copied into your project directory)



# Project Settings cont..

The screenshot shows the STM32CubeMX software interface with the 'Project' tab selected. On the left, there's a sidebar with 'Code Generator' highlighted by a red box. The main area has tabs for 'Pinout & Configuration' and 'Clock Configuration', with 'Clock Configuration' currently active. In the 'Pinout & Configuration' section, under 'STM32Cube MCU packages and embedded software packs', there are three options:

- 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

A pink arrow points from the text '3. Enable “Copy only the necessary files”' to the second option. Below this section, there are sections for 'Generated files' and 'HAL Settings', each containing several checkboxes.

**3. Enable “Copy only the necessary files”**

# Generate Code

Home > STM32U575ZITxQ - NUCLEO-U575ZI-Q > ST67W6X\_BLE\_Commissioning.ioc - Project Manager > LPBAM Scenario & Configuration > GENERATE CODE > CHECK LPE

Pinout & Configuration | Clock Configuration | Project Manager (highlighted) | Tools

Project

Code Generator

Advanced Settings (highlighted)

Driver Selector—  
Search (Ctrl+F) CORTEX\_M33\_NS  
PWR HAL  
RCC HAL  
GPIO HAL  
> GPDMA HAL  
ICACHE HAL  
> SPI HAL  
> USART HAL  
> LPTIM HAL  
STMicroelectronics.X-CUBE-ST67W61.1.0.0 HAL

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance Name	Do Not Generate Function Call	Visibility (Static)
<input checked="" type="checkbox"/>	1	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	2	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	3	MX_GPDMA1_Init	GPDMA1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	4	MX_ICACHE_Init	ICACHE	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	5	MX_SPI1_Init	SPI1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	6	MX_USART1_UART_Init	USART1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	7	MX_LPTIM1_Init	LPTIM1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	10	MX_App_BLE_Commissioning_Init	STMicroelectronics.X-CUBE-ST67W61.1.0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>

4. Enable this option  
(here: we are not generating this function call)

# STM32CubeIDE

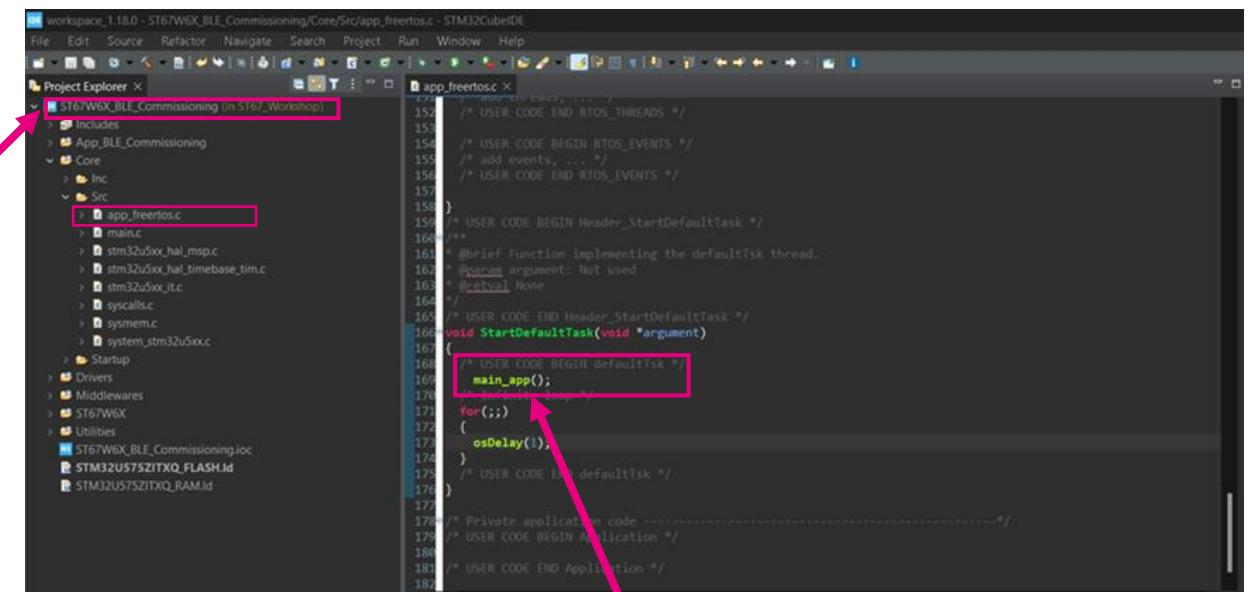
Name	Date modified	Type	Size
App_BLE_Commissioning	7/10/2025 12:05 PM	File folder	
Core	7/10/2025 12:06 PM	File folder	
Drivers	7/10/2025 12:06 PM	File folder	
Middlewares	7/10/2025 12:06 PM	File folder	
ST67W6X	7/10/2025 12:05 PM	File folder	
Utilities	7/10/2025 12:06 PM	File folder	
.cproject	7/10/2025 12:06 PM	CPROJECT File	35 KB
.mxproject	7/10/2025 12:06 PM	MXPROJECT File	17 KB
.project	7/10/2025 12:06 PM	PROJECT File	2 KB
ST67W6X_BLE_Commissioning.ioc	7/10/2025 12:05 PM	STM32CubeMX	21 KB
STM32U575ZITXQ_FLASH.ld	7/10/2025 12:06 PM	LD File	5 KB
STM32U575ZITXQ_RAM.ld	7/10/2025 12:06 PM	LD File	5 KB

1. Load the project on the STM32CubeIDE

After, generating the code, you would view similar folder structure under ST67\_Workshop

```
/* Includes --  
#include "app_freertos.h"  
#include "main_app.h"  
  
/* Private includes --  
/* USER CODE BEGIN Includes */
```

2. Include main\_app.h in app\_freertos.c file  
#include "main\_app.h"



The screenshot shows the STM32CubeIDE interface. The Project Explorer on the left lists the project structure, including the selected 'ST67W6X\_BLE\_Commissioning' folder. The code editor on the right displays the 'app\_freertos.c' file. A pink arrow points from the 'main\_app.h' include in the code editor to the 'main\_app.h' file in the Project Explorer. Another pink arrow points from the 'StartDefaultTask' call in the code editor to the corresponding line in the file.

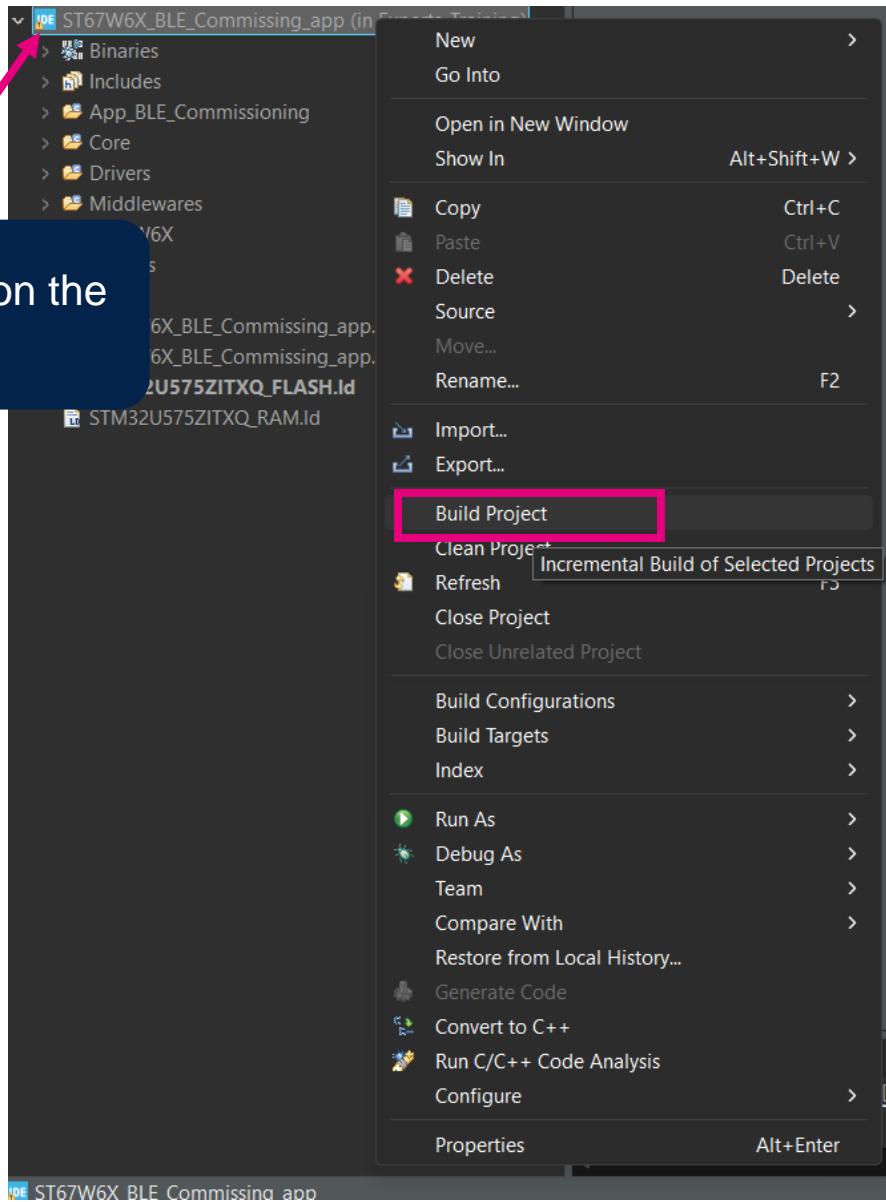
```
159 /* USER CODE BEGIN Header_StartDefaultTask */  
160 /*  
161 * Brief Function implementing the defaultTask thread.  
162 * @param argument: Not used  
163 * @retval None  
164 */  
165 /* USER CODE END Header_StartDefaultTask */  
166 void StartDefaultTask(void *argument)  
{  
    /* USER CODE BEGIN defaultTask */  
    main_app();  
    /* USER CODE END defaultTask */  
    for(;;)  
    {  
        osDelay(1);  
    }  
    /* USER CODE END defaultTask */  
}  
177  
178 /* Private application code --  
179 /* USER CODE BEGIN Application */  
180  
181 /* USER CODE END Application */  
182 }
```

3. Call main\_app() function under StartDefaultTask() in app\_freertos.c file



# Time to build

Right click on the project



```
CDT Build Console [ST67W6X_BLE_Commissioning]
arm-none-eabi-gcc -mcpu=cortex-m33 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8 -c ./Core/main.c
arm-none-eabi-gcc ./Core/Src/app_freertos.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./Core/Src/main.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./Core/Src/stm32u5xx_hal_msp.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./Core/Src/stm32u5xx_hal_timebase_tim.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./Core/Src/stm32u5xx_it.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./Core/Src/syscalls.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./Core/Src/sysmem.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./Core/Src/system_stm32u5xx.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./App_BLE_Commissioning/Target/freertos_tickless.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./App_BLE_Commissioning/Target/logshell_ctrl.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./App_BLE_Commissioning/Target/stm32_lpm_if.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc ./App_BLE_Commissioning/App/main_app.c -mcpu=cortex-m33 -std=gnu11 -g3 -DDEBUG -DconfigTASK_NOTIFICATION_ENTRIES=8
arm-none-eabi-gcc -o "ST67W6X_BLE_Commissioning.elf" @"objects.list" -mcpu=cortex-m33 -T"C:\ST67_Work\ST67W6X_BLE_Commissioning\Linker\linker.ld"
Finished building target: ST67W6X_BLE_Commissioning.elf

arm-none-eabi-size ST67W6X_BLE_Commissioning.elf
arm-none-eabi-objdump -h -S ST67W6X_BLE_Commissioning.elf > "ST67W6X_BLE_Commissioning.list"
    text      data      bss      dec      hex filename
 143952       328   208016   352296   56028 ST67W6X_BLE_Commissioning.elf
Finished building: default.size.stdout

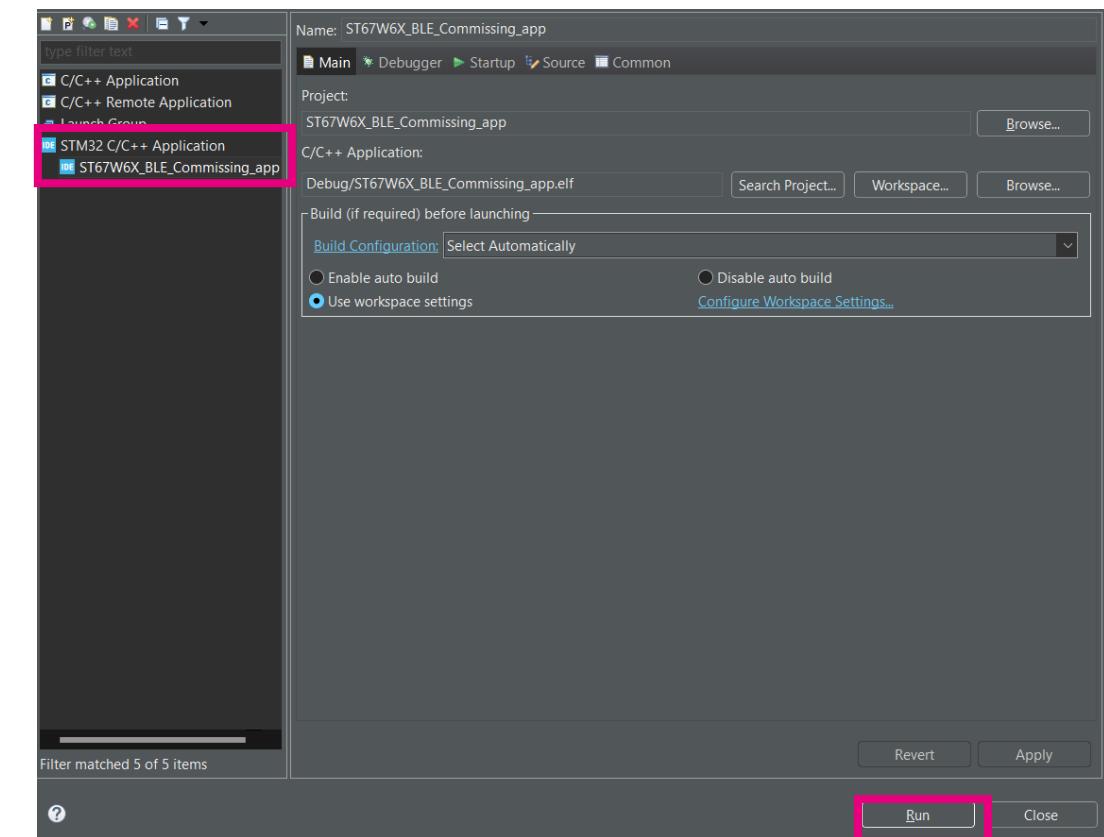
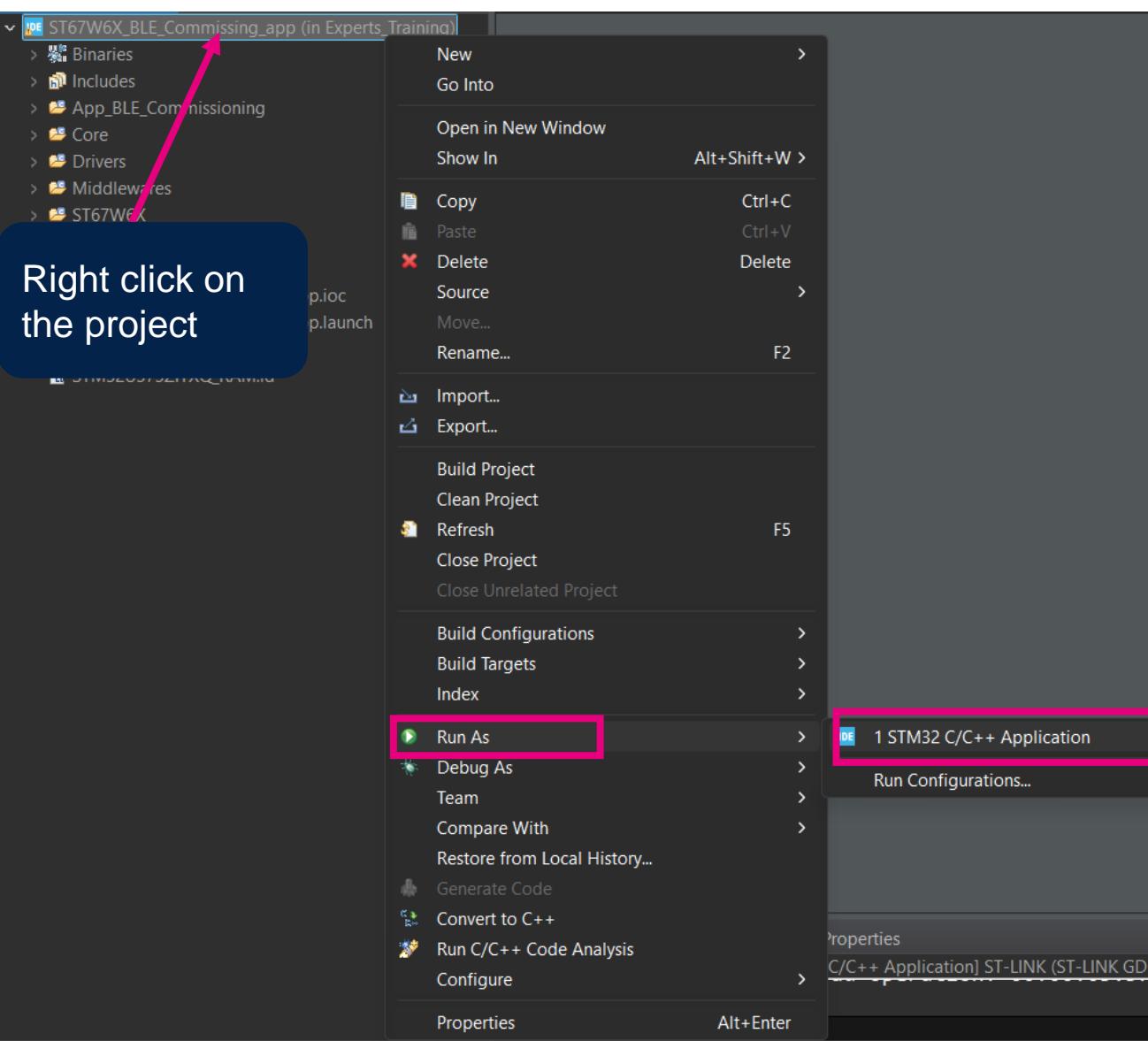
Finished building: ST67W6X_BLE_Commissioning.list

08:12:05 Build Finished. 0 errors, 0 warnings. (took 14s.260ms)
```

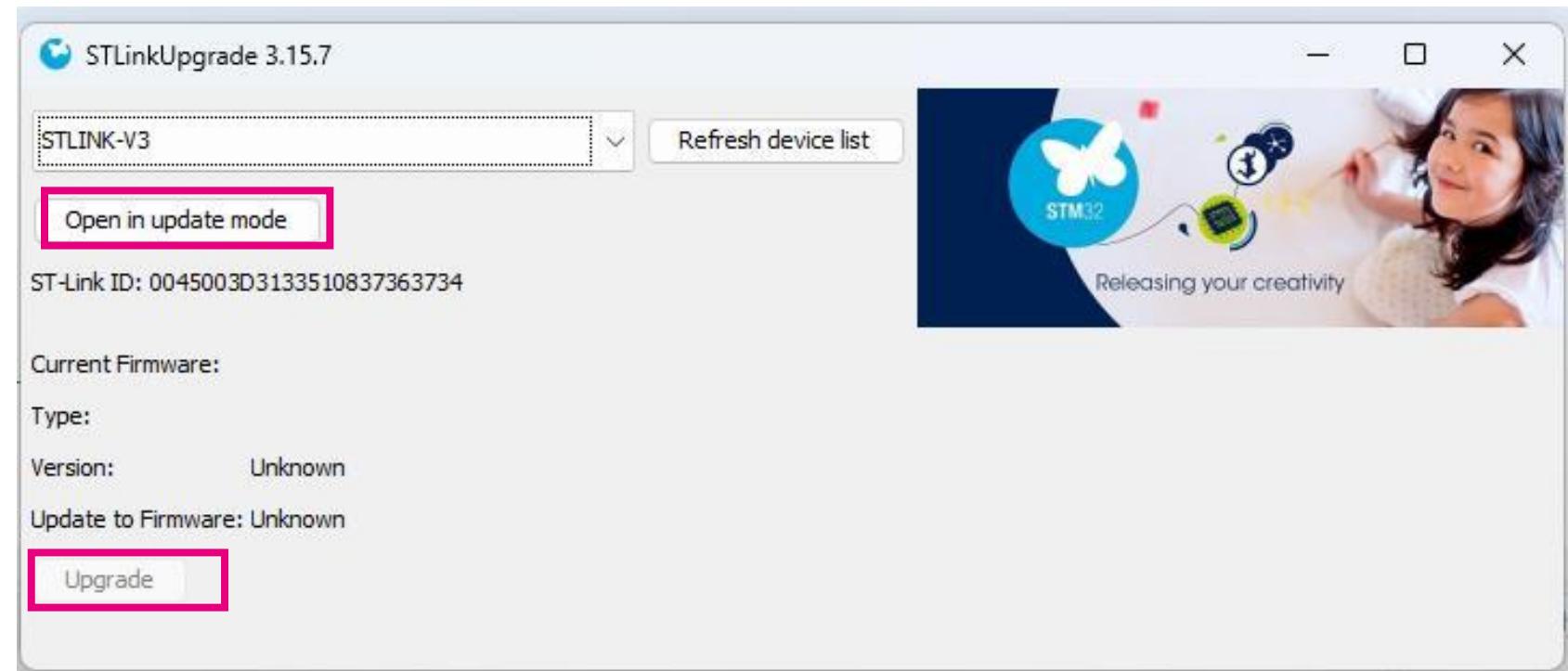
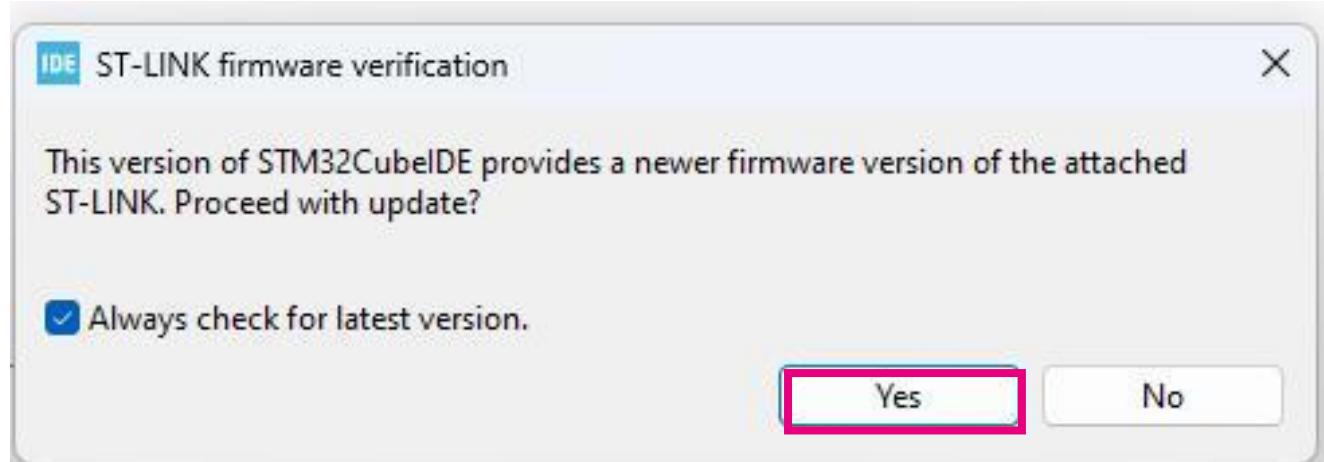
Console after building the application



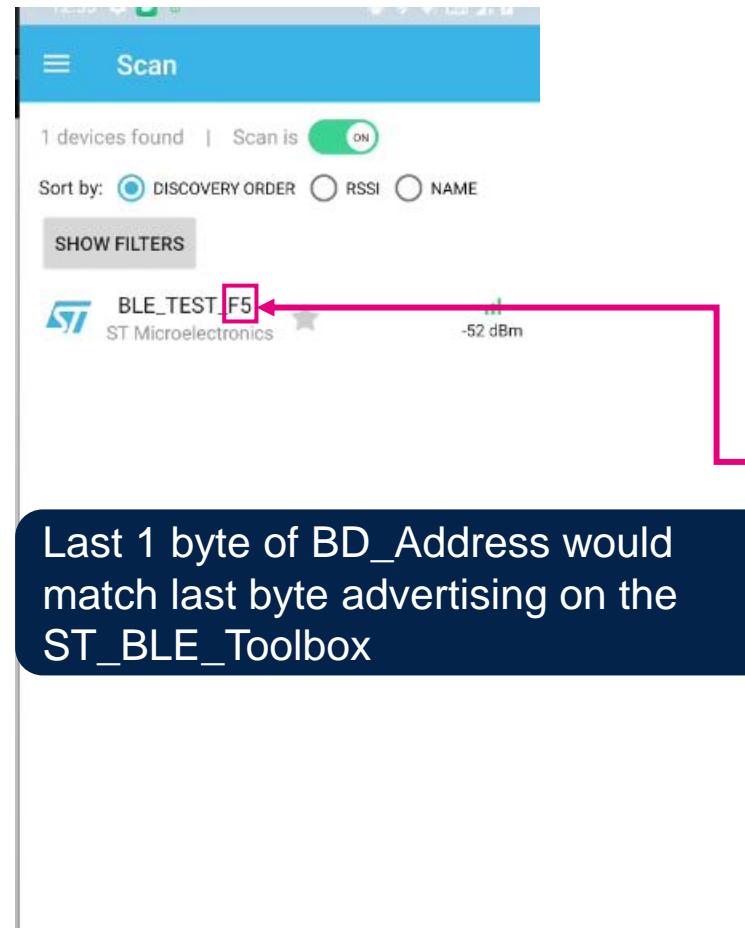
# Flash the project



# ST-LINK Update Warning

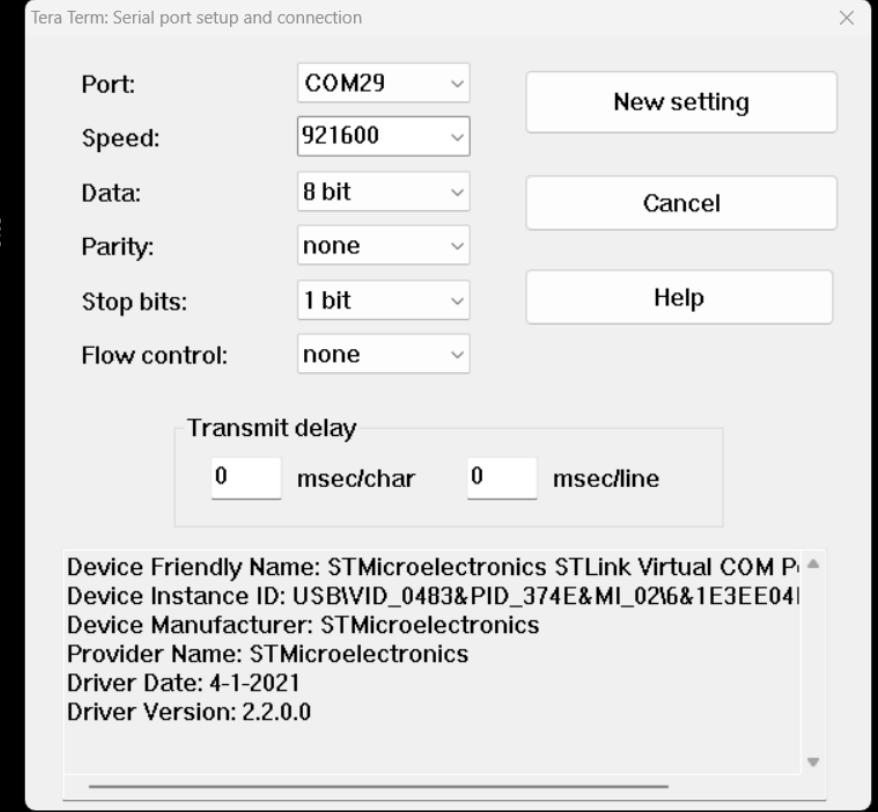


# Tera Term Logs



Last 1 byte of BD\_Address would  
match last byte advertising on the  
ST\_BLE\_Toolbox

```
File Edit Setup Control Window Help
##### Welcome to ST67W6X Wi-Fi Commissioning over BLE Application #####
# build: 11:15:13 Aug 27 2025
---- Host info -----
Host FW Version: 1.0.0
ST67W6X info
ST67W6X MW Version: 1.0.0
AT Version: 1.0.0.1
SDK Version: 2.0.75
MAC Version: 1.6.38
Build Date: May 20 2025 23:01:38
Module ID: C6AFDBBD111400004
BOM ID: 1
Manufacturing Year: 2024
Manufacturing Week: 47
Battery Voltage: 3.320 V
Irin Wi-Fi hp: 2.2,2.2,2.2,2.2,2.2,2.2,2.1,2
Irin Wi-Fi lp: 2.3,3.4,4.5,5.5,5.5,5.5,5.5,5.5
Irin BLE: 1.2,2.1,1
Irin XTAL: 40
MAC Address: 40:82:7b:00:12:f4
Anti-rollback Bootloader: 0
Anti-rollback App: 0
Wi-Fi init is done
Net init is done
Bla_init is done
BD Address: 40:82:7b:00:12:f5
Configure BLE
BLE configuration is done
BLE Commissioning Service Creation
- BLE service created
- BLE WIFI Control charac created
- BLE WIFI Configure charac created
- BLE WIFI Monitoring charac created
- BLE services and charac registered
BLE service and charac creation is done
Start BLE advertising
BLE advertising is started
```



Tera Term logs after flashing the BLE  
Commissioning application



# ST BLEToolbox/ ST BLE Webpage

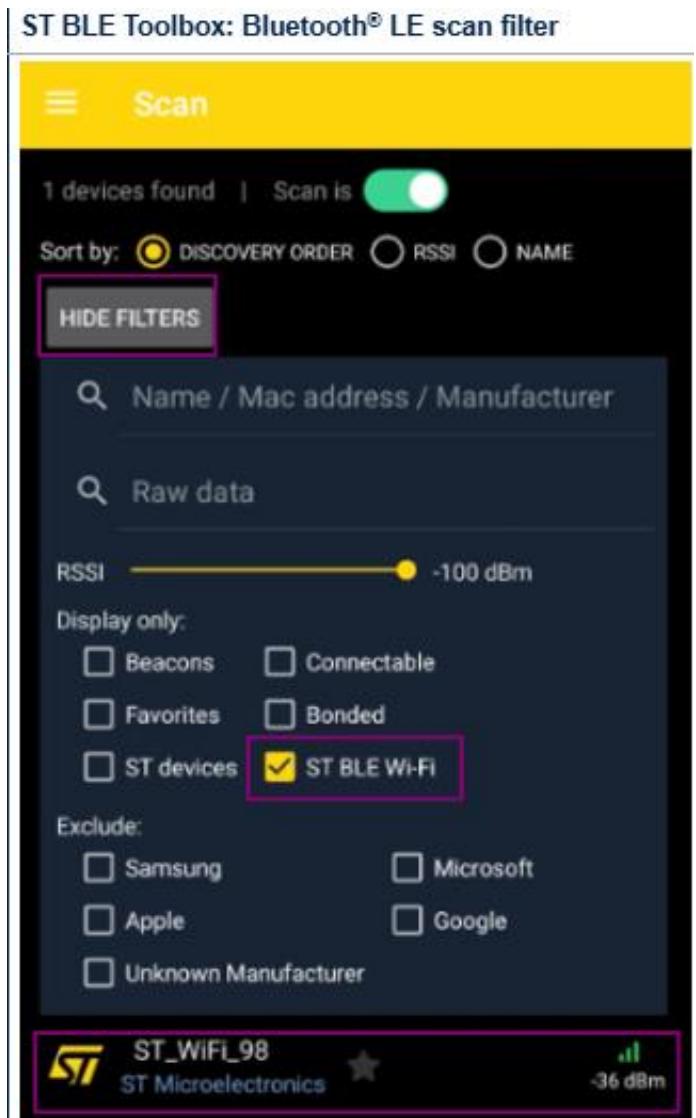
To interface with the Wi-Fi commissioning embedded application, you can use  
[STBLEToolbox Android/iOS application](#) / [ST Web Bluetooth® Interface](#)

Download [STBLEToolBox](#)

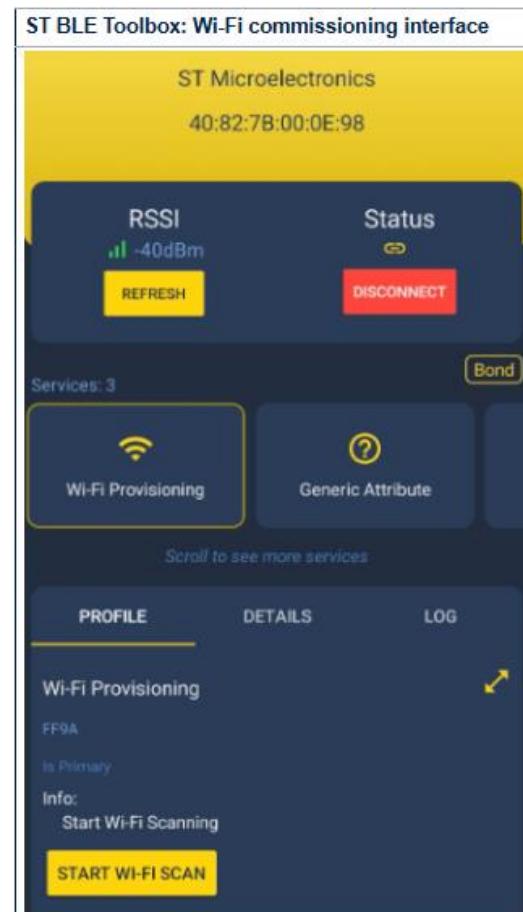
The versions supporting Wi-Fi commissioning service  
are **v1.4.3 for Android and 1.2.8 for iOS**

## Steps to launch and connect to Wi-Fi AP:

1. Launch the smartphone application and enable the scan button, if it is not already activated.
2. You can filter scanned devices by clicking on the SHOW FILTERS button and select ST BLE Wi-Fi to show only devices with Wi-Fi commissioning service.
3. Select your device, named ST\_WiFi\_XX where XX represents the last BD address digit, and connect to it.



4. Once connected to the Bluetooth® LE device, click on Wi-Fi commissioning button to open the commissioning dedicated interface.



From this interface,  
Launch a Wi-Fi scan (1),  
To detect and list all available networks(2).  
Once the available networks are displayed, select the one you want to connect to (2)  
Type a password if needed (3)  
Click on CONNECT button (4).

A sequence of four screenshots illustrating the Wi-Fi connection process:

- The first screenshot shows the 'Wi-Fi connection interface' with the 'START WI-FI SCAN' button highlighted (1).
- The second screenshot shows the results of the scan, listing available networks: 'AndroidAP22A1' (Security: WPA2), 'Ruc-TestBench' (Security: WPA2), 'Zyxel\_6A4D' (Security: WPA2), and 'STWLAN2' (Security: WPA\_Enterprise). The 'AndroidAP22A1' entry is selected (2).
- The third screenshot shows the 'Wi-Fi Provisioning' screen with a password field containing '3' (3).
- The fourth screenshot shows the final step where the 'CONNECT' button is highlighted (4).



Once the connection request is done, the connection status is displayed in the interface.  
Clicking on the Ping button start a ping test with the Wi-Fi access point and returns results in the interface.

The image shows two screenshots of the ST-Link V3 interface. The left screenshot displays the 'Wi-Fi status and ping' screen for a device named 'ST\_WIFI\_98'. It shows an RSSI of -40dBm, a 'Status' button, and a 'Bond' button. Below these are sections for 'Wi-Fi Provisioning' and 'Generic Attribute'. At the bottom, there are tabs for 'PROFILE', 'DETAILS', and 'LOG', with 'Wi-Fi Provisioning' selected. The right screenshot shows the 'Wi-Fi Provisioning' screen for a device with MAC address '40:82:7B:00:0E:98'. It includes an 'RSSI -40dBm' section, a 'Status' button, a 'Bond' button, and sections for 'Wi-Fi Provisioning' and 'Generic Attribute'. A yellow arrow points from the 'LOG' tab in the first screenshot to the log window on the right. The log window displays a series of BLE events and WiFi AP scan results. The log starts with:

```

BLE service and charac creation is done
Start BLE advertising
BLE advertising is started
BLE connected
-> BLE CONNECTED: Conn_Handle: 0
BLE MTU exchange
BLE notification enabled
-> BLE NOTIFICATION ENABLED [Service: 0, Charac: 2].
BLE write
-> BLE WRITE.
-> Conn_Handle: 0, Service: 0, Charac: 0, length 1, Data:
0x01
WIFI Control Charac
WIFI Scan Enable
WIFI Scan done.

```

The log continues with a large list of WiFi AP scan results, each entry containing MAC address, Channel, RSSI, SSID, and other details. The log ends with:

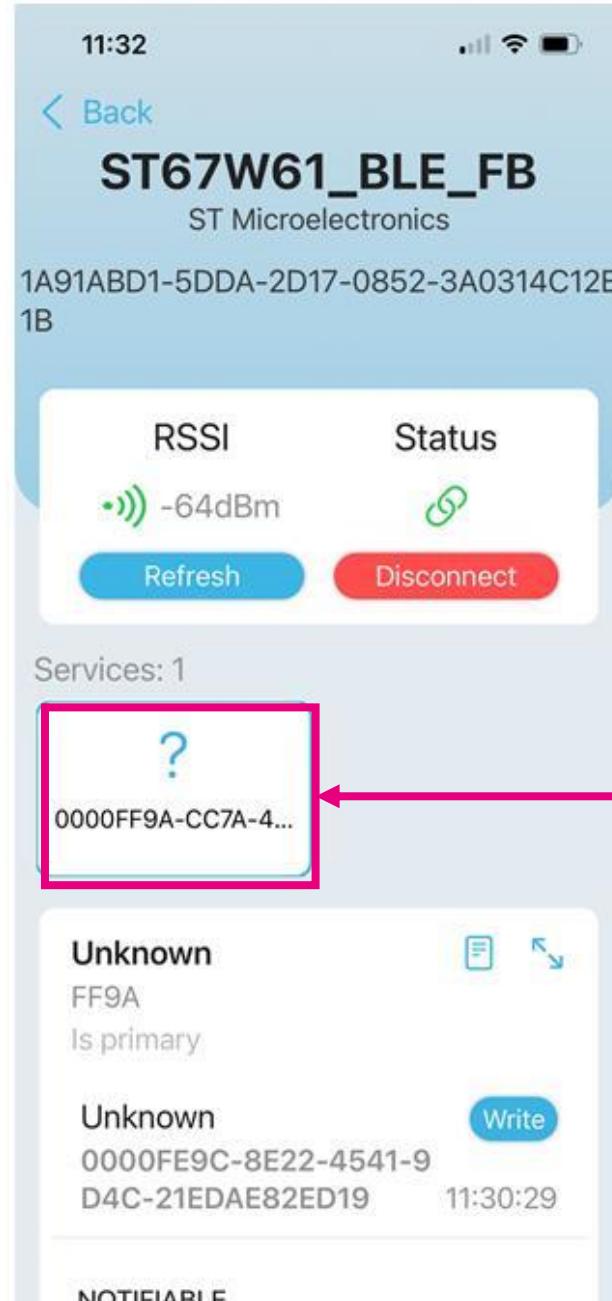
```

WPA-EAP | RSSI: -73 | SSID: DIRECT-4A-HP DeskJet
WPA2 | RSSI: -77 | SSID: XFWHG
OPEN | RSSI: -78 | SSID: SmartLife-9EAD

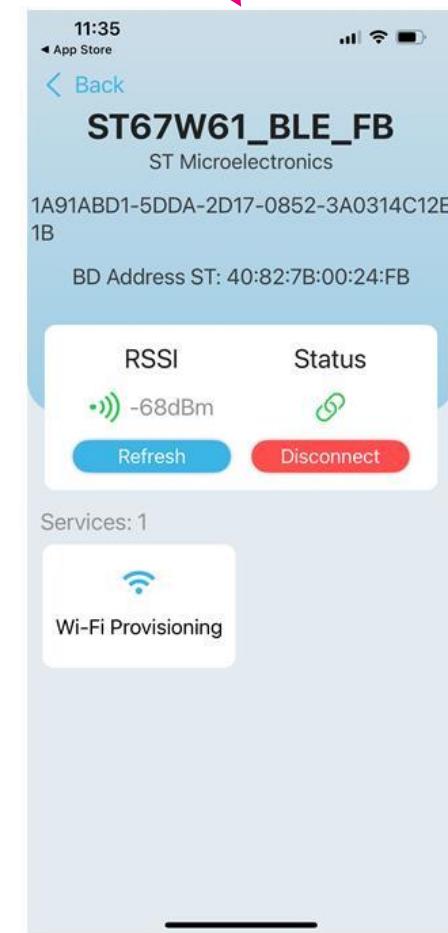
```



# ST BLE Toolbox



Solution: Uninstall the app and re-install the app back





# Porting to a new host mcu using STM32CubeMX



# Porting over new host mcu(1/3)

## 1. Select the Host MCU

Choose the target STM32 part number from the STM32CubeMX

## 2. Check schematics of Host MCU and ST67

Review pin mappings for SPI, GPIO, USART

## 3. Configure USART for Debugging

Used for debugging logs via STLink virtual port

## 4. Configure the SPI Interface for Host MCU and ST67 communication

Set SPI pins, SPI mode, SPI Priority

## 5. Configure DMA Channels for SPI

Fast and reliable SPI data transfer, reducing CPU load during transmission



# Porting over new host mcu(2/3)

## 6. Configure GPIO Pins ( CHIP\_EN, SPI\_RDY, SPI\_CS, BOOT, User\_Button)

Set GPIO Mode, GPIO output level, GPIO Maximum Output speed

## 7. Configure SYS (Time base Source)

## 8. Configure Clock

## 9. Enable FreeRTOS Middleware

Setup default task, heap size, stack size and required priorities

## 10. Enable X-CUBE-ST67W61

Setup Application, W61 Drivers, Utilities

## 11. Configure NVIC Interrupt Settings

SPI, DMA, USART, EXTI



# Porting over new host mcu(3/3)

## 12. Project Creation and Code Generation

Generate project files (STM32CubeIDE/IAR)

Open in IDE for application development

## 13. Application Entry: Call main\_app()

Inside StartDefaultTask, invoke main\_app()

This initializes ST67W6X Driver, ST67W6X Wi-Fi module, ST67W6X Network module, Application specific functions

## 14. Build and Flash the application

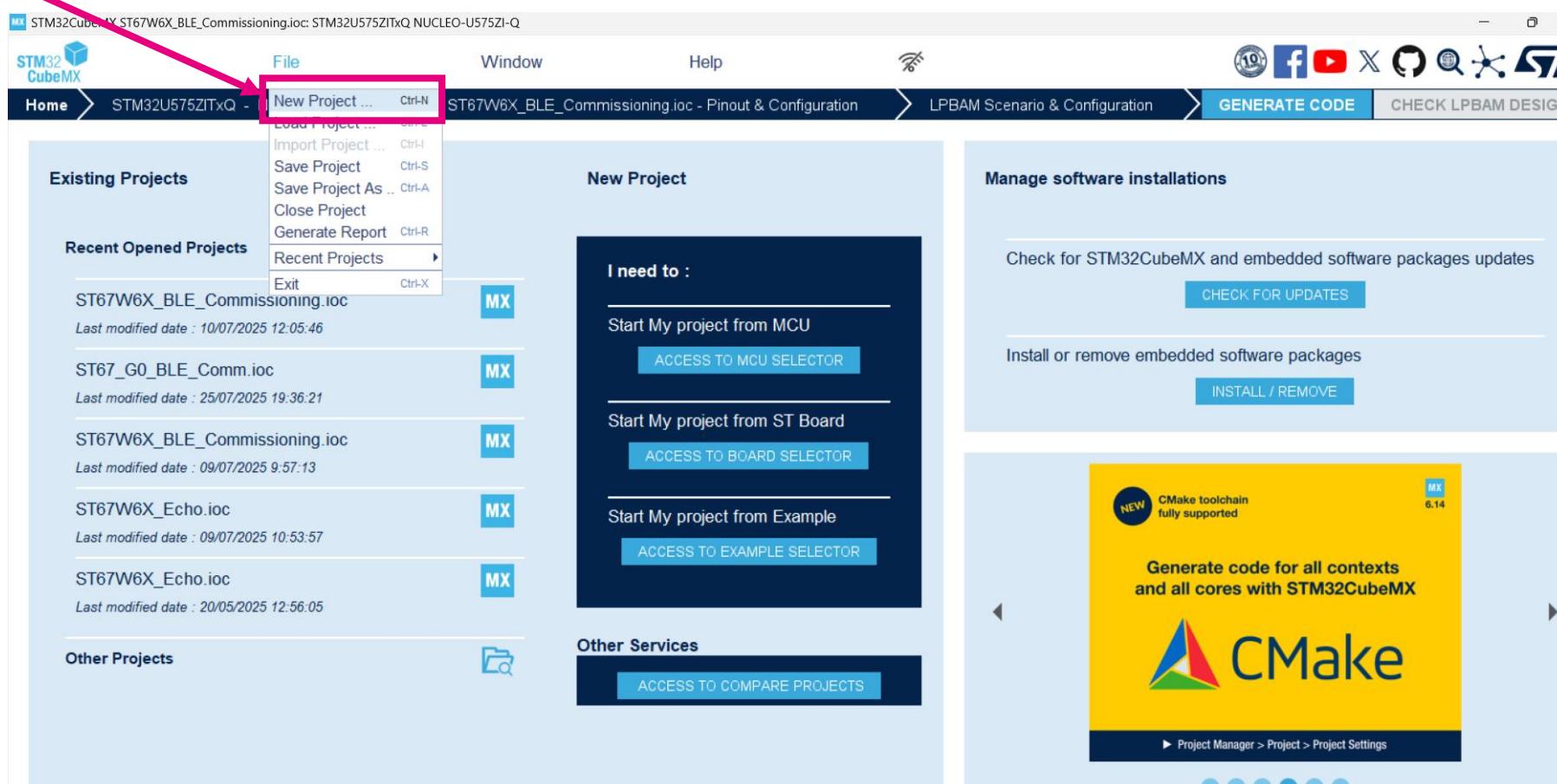




## Generating BLE Commissioning app using NUCLEO-G0B1RE as host processor (Hands-On)

# Case 3: Generating BLE Commissioning app using NUCLEO-G0B1RE as host processor

Open STM32CubeMX and create the new project.



Select the board selector

The screenshot shows the STBoard Selector interface. On the left, a sidebar lists STM32 series: STM32C0, STM32F0, STM32F1, STM32F2, STM32F3, STM32F4, STM32F7, STM32G0, STM32G4, STM32H5, STM32I7, STM32L0, STM32L1, STM32L4, STM32L4+, STM32L5, STM32MP1, STM32MP2, STM32N6, and STM32U0. The 'STM32G0' option is selected and highlighted with a red box. At the top, tabs include 'MCU/MPU Selector' (selected), 'Board Selector' (highlighted with a red box), 'Example Selector', and 'Cross Selector'. A search bar contains 'STM32G0'. Below the sidebar is a 'PRODUCT INFO' section with a dropdown menu showing 'STM32G0' again. In the center, the 'NUCLEO-G0B1RE' product page is displayed. It features a large image of the board, a brief description, and a 'Start Project' button highlighted with a red box. The 'NUCLEO-G0B1RE' row is also highlighted with a red box in a table below.

Thumbnail	Commercial Part No.	Type	Marketing Status	Unit Price (US\$)	Mounted Device
	NUCLEO-G0B1RE	Nucleo-64	Active	10.32	STM32G0B1RET6
	NUCLEO-G031KB	Nucleo-32	Active	10.32	STM32G031KBT6
	NUCLEO-G070RB	Nucleo-64	Active	10.32	STM32G070RBT6
	NUCLEO-G071RB	Nucleo-64	Active	10.32	STM32G071RBT6

Type STM32G0

Select STM32G0

Start Project

Select NUCLEO-G0B1RE



MCU/MPU Selector | Board Selector | Example Selector | Cross Selector

Board Filters

- Commercial
- 
- 
- 

PRODUCT INFO

- Type >
- Supplier >
- MCU / MPU Series >

Aa [ab]

- STM32C0
- STM32F0
- STM32F1
- STM32F2
- STM32F3
- STM32F4
- STM32F7
- STM32G0
- STM32G4
- STM32H5
- STM32H7
- STM32L0
- STM32L1
- STM32L4
- STM32L4+
- STM32F5

Features

Large Picture

Docs & Resources

Datasheet

Buy

Start Project

STM32G0 Series

**NUCLEO-G0B1RE**

**STM32 Nucleo-64 development board with STM32G0B1RE MCU, supports Arduino and ST morpho connectivity**

**ACTIVE**  
Product is in mass production

Part Number : NUCLEO-G0B1RE  
Commercial Part Number : NUCLEO-G0B1RE  
Unit Price (US\$) : 10.32  
Mounted Device : STM32G0B1RET6

The STM32 Nucleo-64 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode.

DUINO® Uno V3 connectivity support and the ST morpho headers allow the easy expansion of the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields.

STM32 Nucleo-64 board does not require any separate probe as it integrates the

Board Project Options: NUCLEO-G0B1RE

Initialize all peripherals with their default Mode ?

Yes No

Bands List: 8 items

	Overview	Commercial Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
<input checked="" type="checkbox"/>		NUCLEO-G070RB	Nucleo-64	Active	10.32	STM32G070RBT6
<input checked="" type="checkbox"/>		NUCLEO-G071RB	Nucleo-64	Active	10.32	STM32G071RBT6
<input checked="" type="checkbox"/>		NUCLEO-G0B1RE	Nucleo-64	Active	10.32	STM32G0B1RET6
<input checked="" type="checkbox"/>		STM32G0316-DISCO	Discovery Kit	Active	9.89	STM32G0316RM6
<input checked="" type="checkbox"/>		NUCLEO-F030R8	Nucleo-F030R8	Active	10.32	STM32F030R8T6
<input checked="" type="checkbox"/>		NUCLEO-F030K6	Nucleo-F030K6	Active	10.32	STM32F030K6T6
<input checked="" type="checkbox"/>		NUCLEO-F030C6	Nucleo-F030C6	Active	10.32	STM32F030C6T6

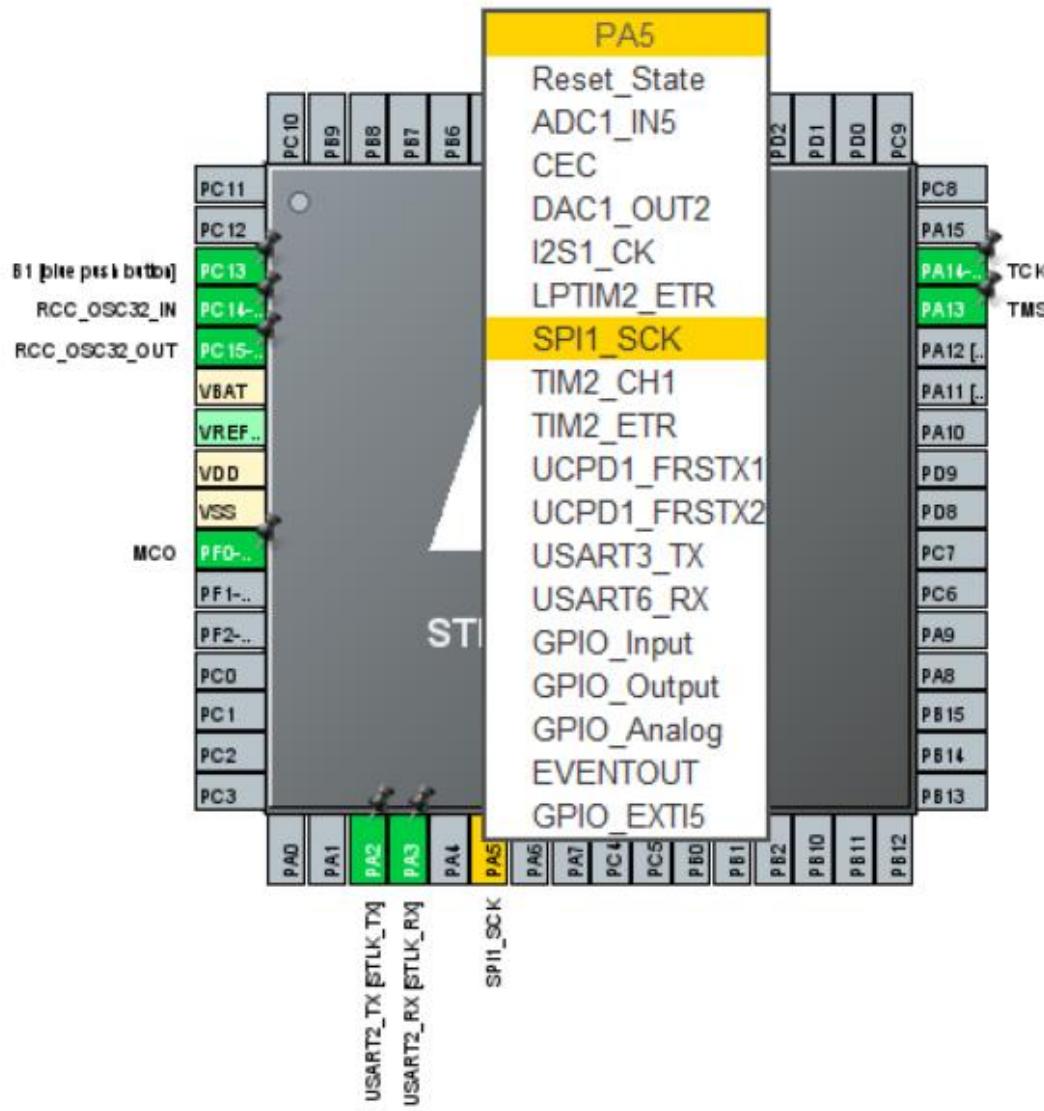
Select Yes

# Pinout

When generating an application from scratch and if X-Nucleo-67W61M1 is used, the pinout must be set in accordance with the ST67W61M1 Arduino interfaces.

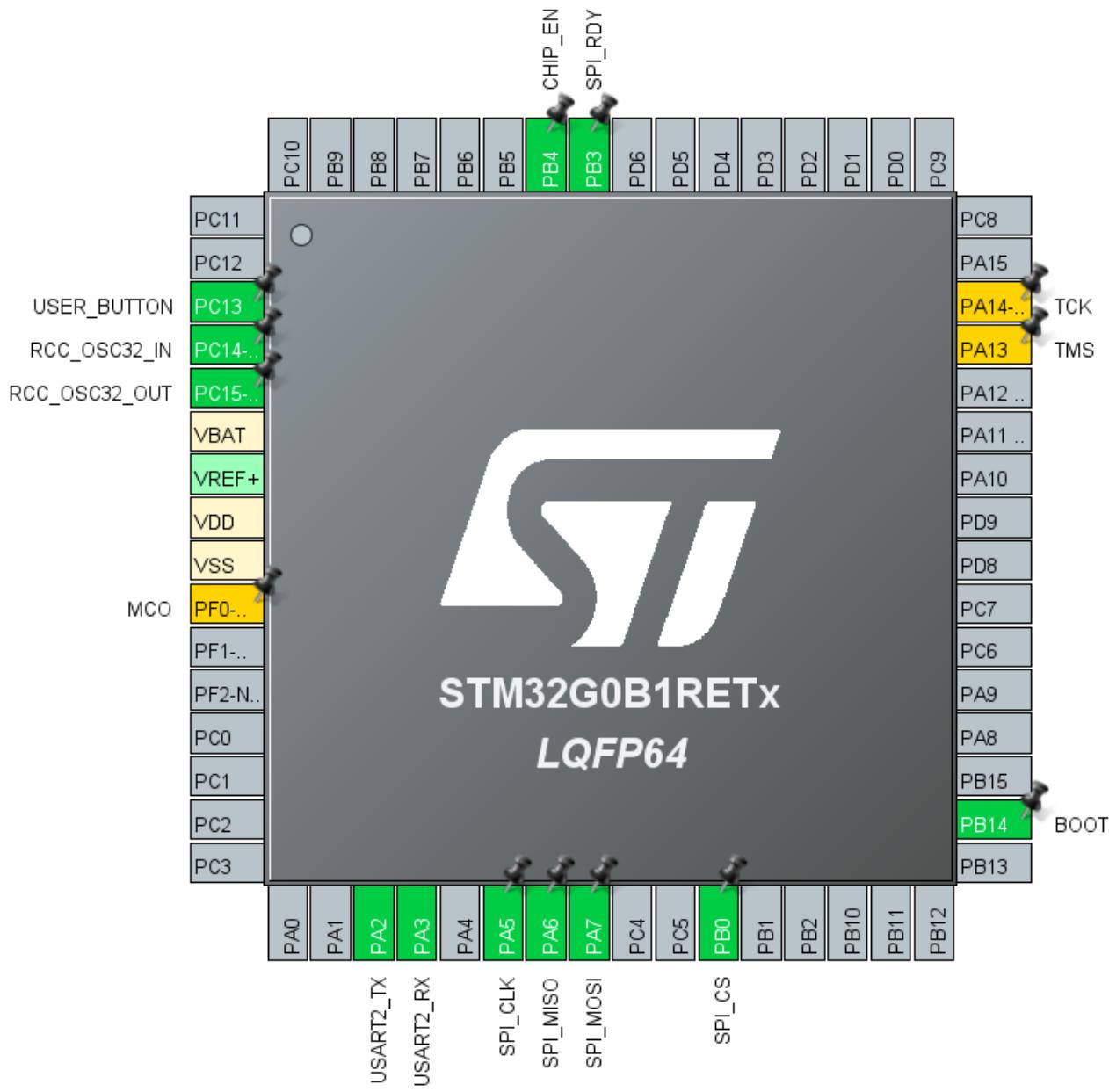
Pin function	Arduino Connector	STM32G0 Pin	GPIO mode	GPIO pull mode	GPIO speed
SPI_CLK	CN5.D13	PA5	AF PP	No Pull	Very high
SPI_MISO	CN5.D12	PA6	AF PP	No Pull	Very high
SPI_MOSI	CN5.D11	PA7	AF PP	No Pull	Very high
SPI_CS	CN5.D10	PB0	Output PP	No Pull	high
USER_BUTTON	-	PC13	EXTI Falling	Pull-up	N/A
CHIP_EN	CN9.D5	PB4	Output PP	No Pull	High
BOOT	CN9.D6	PB14	Output PP	No Pull	Low
SPI_RDY	CN9.D3	PB3	EXTI Falling/Rising	No Pull	N/A

# Assigning functionality



For assigning, click on the pin and select functionality.  
(e.g) PA5->SPI\_SCK

# Pinout View



After, assigning all the pins mentioned from the pin out table, this would be overall pinout view



# UART GPIO(USART 2)

The screenshot shows the ST-LINK V3 Configuration interface for a STM32 microcontroller. The main window title is "Parameter Settings". A pink arrow points from a callout box "Select USART2" to the USART2 entry in the left sidebar, which is highlighted with a blue background. Another pink arrow points from a callout box "GPIO Settings" to the "GPIO Settings" tab in the top navigation bar of the configuration window.

**Select USART2**

**Parameter Settings**

**GPIO Settings**

**USART2 TX: PA2**

**USART2 RX: PA3**

**Configuration**

**Mode:** Asynchronous

**Hardware Flow Control (RS232):** Disable

**Slave Select(NSS) Management:** Disable

**Configuration**

**Reset Configuration**

**Parameter Settings** (selected)

**User Constants**

**NVIC Settings**

**DMA Settings**

**GPIO 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**

- Data Direction: Receive and Transmit
- Over Sampling: 16 Samples
- Single Sample: Disable
- ClockPrescaler: 1
- Fifo Mode: Disable
- Tx fifo Threshold: 1 eighth full configuration
- Rx fifo Threshold: 1 eighth full configuration

**Advanced Features**

- Auto Baudrate: Disable
- TX Pin Active Level Inversion: Disable
- RX Pin Active Level Inversion: Disable
- Data Inversion: Disable
- TX and RX Pins Swapping: Disable
- Overrun: Enable
- DMA on RX Error: Enable
- MSB First: Disable

**Configuration**

**Reset Configuration**

**Parameter Settings** (selected)

**User Constants**

**NVIC Settings**

**DMA Settings**

**GPIO Settings**

**Search Signals**

**Search (Ctrl+F)**

**Show only Modified Pins**

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum out...	Fast Mode	User Label	Modified
PA2	USART2_TX	Low	Alternate Function	No pull-up and no pull-down	Low	n/a		
PA3	USART2_RX	Low	Alternate Function	No pull-up and no pull-down	Low	n/a		

**ST**

# SPI GPIO(SPI1)

PA5: SPI\_SCK  
PA6: SPI\_MISO  
PA7: SPI\_MOSI

Select SPI1

Pinout & Configuration

Clock Configuration

Project Manager

Mode: Full-Duplex Master

Hardware NSS Signal: Disable

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output	Fast Mode	User Label	Modified
PA5	SPI1_SCK	Low	Alternate Function	No pull-up and no pull-down	Very High	n/a	SPI_SCK	✓
PA6	SPI1_MISO	Low	Alternate Function	No pull-up and no pull-down	Very High	n/a	SPI_MISO	✓
PA7	SPI1_MOSI	Low	Alternate Function	No pull-up and no pull-down	Very High	n/a	SPI_MOSI	✓



# SPI GPIO(SPI1)

The screenshot shows the STM32CubeMX software interface for configuring the SPI1 peripheral. A pink arrow points from a 'Parameter Settings' callout to the 'Parameter Settings' tab in the top navigation bar. Another pink arrow points from a 'Set the Data size to 8' callout to the 'Data Size' field set to 8 Bits. A third pink arrow points from a 'Set the Prescaler to 2' callout to the 'Prescaler (for Baud Rate)' field set to 2.

**Parameter Settings**

**SPI1 Mode and Configuration**

Mode: Full-Duplex Master  
Hardware NSS Signal: Disable

**Configuration**

Reset Configuration

Parameter Settings (highlighted with a red box)

User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

Search (Ctrl+F) | Back | Forward | i

**Basic Parameters**

- Frame Format
- Data Size (highlighted with a red box, set to 8 Bits)
- First Bit

**Clock Parameters**

- Prescaler (for Baud Rate) (highlighted with a red box, set to 2)
- Baud Rate: 8.0 MBits/s
- Clock Polarity (CPOL): Low
- Clock Phase (CPHA): 1 Edge

**Advanced Parameters**

- CRC Calculation: Disabled
- NSSP Mode: Enabled
- NSS Signal Type: Software

# DMA Channels

Select DMA

The screenshot shows the STM32CubeMX software interface for configuring DMA channels. A pink arrow points from the 'Select DMA' text to the 'DMA' category in the left sidebar.

**Pinout & Configuration** tab is active.

**Clock Configuration** tab is visible in the top right.

**DMA Mode and Configuration** section:

DMA Request	Channel	Direction	Priority
SPI1_TX	DMA1 Channel 1	Memory To Peripheral	High
SPI1_RX	DMA1 Channel 2	Peripheral To Memory	Low Medium High Very High

**DMA Request Settings** section:

- Mode: Normal
- Increment Address: Peripheral (unchecked), Memory (checked)
- Data Width: Byte (Peripheral and Memory)

**DMA Request Synchronization Settings** section:

- Enable synchronization:
- Synchronization signal:
- Signal polarity:
- Enable event:
- Request number:

**SPI1\_TX -> DMA1 CHANNEL 1->High**  
**SPI1\_RX -> DMA1 CHANNEL 2-> High**



# GPIO

## Select GPIO

GPIO Mode and Configuration

Mode

Configuration

Group By Peripherals

GPIO Single Mapped Signals RCC SPI USART NVIC

Search Signals Search (Ctrl+F)

Show only Modified Pins

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pul...	Maximu...	Fast Mode	User Label	Modified
PB0	n/a	Low	Output Push Pull	No pull-u...	High	n/a	SPI_CS	<input checked="" type="checkbox"/>
PB3	n/a	n/a	External Interrupt Mode with Rising/Falling edge trigger detection	No pull-u...	n/a	n/a	SPI_RDY	<input checked="" type="checkbox"/>
PB4	n/a	Low	Output Push Pull	No pull-u...	High	n/a	CHIP_EN	<input checked="" type="checkbox"/>
PB14	n/a	Low	Output Push Pull	No pull-u...	Low	n/a	BOOT	<input checked="" type="checkbox"/>
PC13	n/a	n/a	External Interrupt Mode with Falling edge trigger detection	No pull-u...	n/a	n/a	USER_BU...	<input checked="" type="checkbox"/>

GPIO pin names

GPIO output level

GPIO mode

GPIO maximum speed

GPIO user label

# Timer

SYS Mode and Configuration

Mode

- Serial Wire
- System Wake-Up 1
- System Wake-Up 2
- System Wake-Up 4
- System Wake-Up 5
- System Wake-Up 6

Power Voltage Detector In

VREFBUF Mode

Pin PA9 instead of pin PA11

Pin PA10 instead of pin PA12

Timebase Source   save power of non-active UCPD - deactivate Dead Battery pull-up

Categories A-Z

System Core

- ✓ DMA
- ✓ GPIO
- IWDG
- ✓ NVIC
- ✓ RCC
- SYS**
- WWDG

Analog

Timers

Connectivity

Multimedia

Computing

Sys mode configuration: TIM1 in place  
of Systick: to avoid RTOS conflicts, enable low  
power operation.

Home > STM32G0B1RETx - NUCLEO-G0BIRE > Untitled - Pinout & Configuration >

Pinout & Configuration      Clock Configuration      Project Manager

Categories A-Z

System Core

- ✓ DMA
- ✓ GPIO
- IWDG
- RCC
- RTC
- WWDG

Analog

Timers

Connectivity

- FDCAN1
- FDCAN2
- I2C1
- I2C2
- I2C3
- IRTIM
- LPUART1
- LPUART2

- ✓ SPI1
- SPI2
- SPI3
- UCPD1
- UCPD2
- USART1
- ✓ USART2
- USART3
- USART4
- USART5
- USART6
- USB\_DRD\_FS

Multimedia

Computing

Middleware and Software Packs

Utilities

High Speed Clock (HSE)  Disable

OSC enable

Low Speed Clock (LSE) Crystal/Ceramic Resonator

OSC32 enable

Master Clock Output

Master Clock Output 2

LSCO Clock Output

Audio Clock Input (2S\_CKIN)

CRS SYNC  Disable

**HSE disabled**

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

System Parameters

- VDD voltage (V) 3.3 V
- Instruction Cache Enabled
- Prefetch Buffer Enabled
- Data Cache Enabled
- Flash Latency(WS) 2 WS (3 CPU cycle)

RCC Parameters

- HSI Calibration Value 64
- HSE Startup Timeout Value (ms) 100
- LSE Startup Timeout Value (ms) 5000

Power Parameters

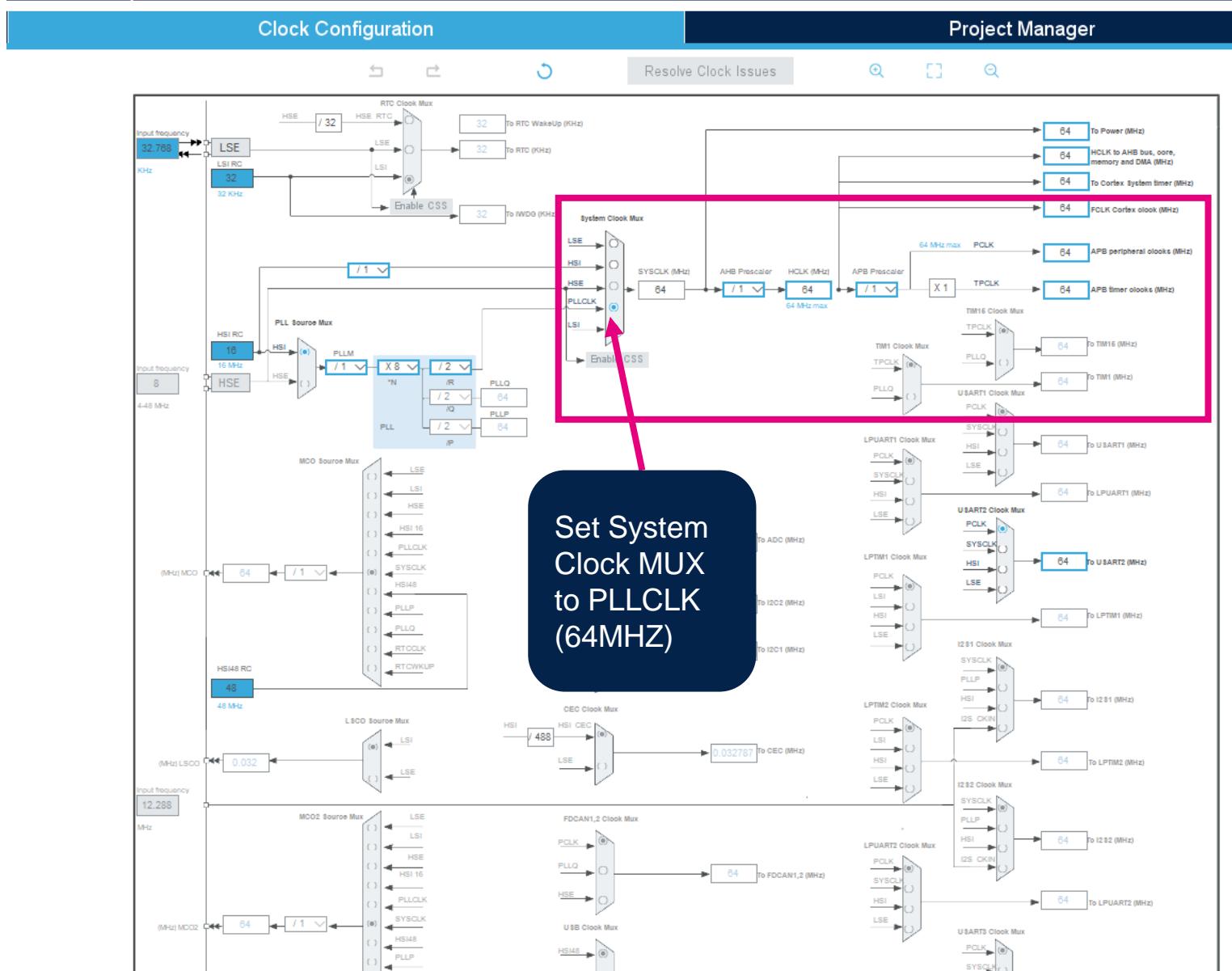
- Power Regulator Voltage Scale Power Regulator Voltage Scale 1

Peripherals Clock Configuration

- Generate the peripherals clock configuration TRUE

# Clock Configuration

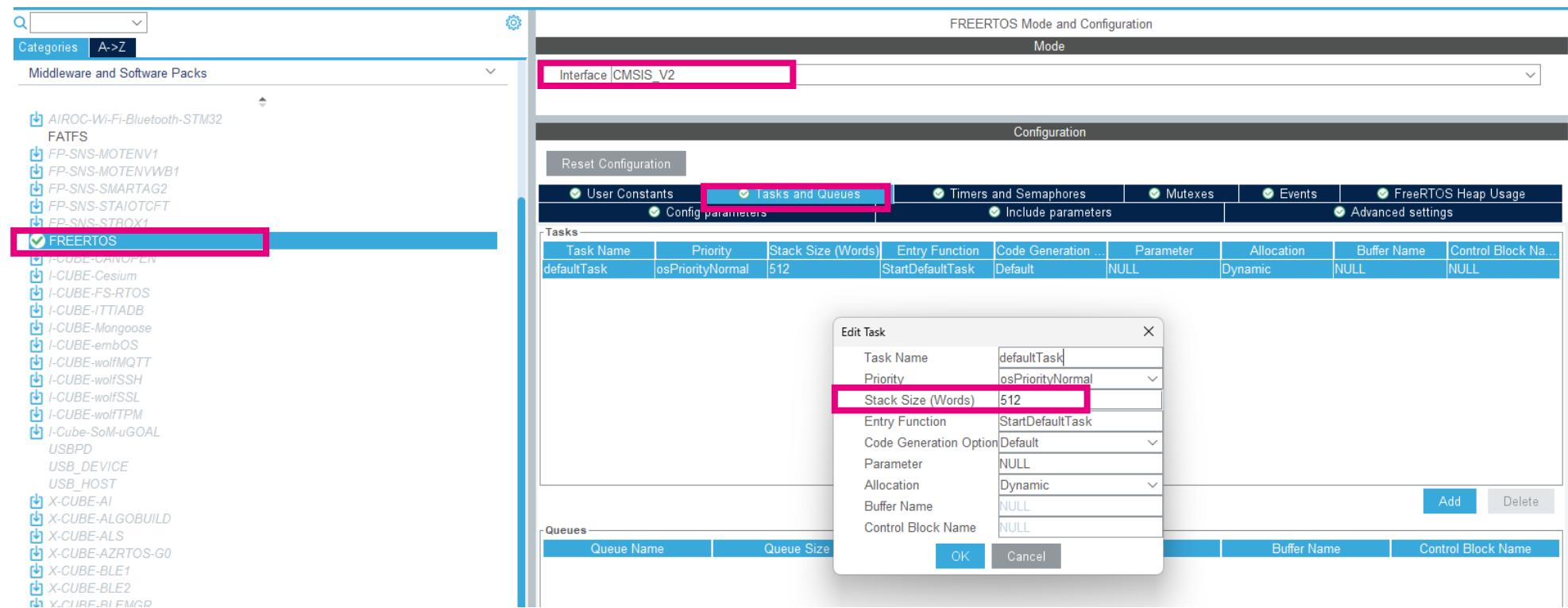
- In the "Clock Configuration" panel, system clock is set to the maximum frequency of 64 MHz, which increases the SPI clock speed to 32 MHz



# Middleware & Software Packs

## FreeRTOS

- FreeRTOS component can be either be selected from Middleware (FREERTOS) or from the Software Pack (X-CUBE-FREERTOS) depending on the chosen STM32.
- By default, the stack size of the default task defined by STM32CubeMx tool is equal to 128 words. The application default task required **512 words**.



# Middleware & Software Packs

## FreeRTOS

- Adjust heap :The size heap value should be set at least to 40Kbytes. To optimize the memory, this value could be adjusted during integration and validation tests of the application

The screenshot shows the ST-Middleware & Software Packs interface. On the left, a sidebar lists various software packs, with 'FREERTOS' selected and highlighted in red. The main panel is titled 'FREERTOS Mode and Configuration' and contains several tabs: 'Mode', 'Configuration', 'Reset Configuration', 'User Constants', 'Tasks and Queues' (which is currently selected and highlighted in red), 'Timers and Semaphores', 'Mutexes', 'Events', and 'FreeRTOS Heap Usage'. Under the 'Configuration' tab, there is a section for 'Configure the below parameters :'. A search bar is available. The configuration parameters are organized into sections: 'MPU/FPU' (ENABLE\_MPUSet to Disabled, ENABLE\_FPU set to Disabled), 'Kernel settings' (USE\_PREEMPTION set to Enabled, CPU\_CLOCK\_HZ set to SystemCoreClock, TICK\_RATE\_HZ set to 1000, MAX\_PRIORITIES set to 56, MINIMAL\_STACK\_SIZE set to 128 Words, MAX\_TASK\_NAME\_LEN set to 16, USE\_16\_BIT\_TICKS set to Disabled, IDLE\_SHOULD\_YIELD set to Enabled, USE\_MUTEXES set to Enabled, USE\_RECURSIVE\_MUTEXES set to Enabled, USE\_COUNTING\_SEMAPHORES set to Enabled, QUEUE\_REGISTRY\_SIZE set to 8, USE\_APPLICATION\_TASK\_TAG set to Disabled, ENABLE\_BACKWARD\_COMPATIBILITY set to Enabled, USE\_PORT\_OPTIMISED\_TASK\_SELECTION set to Disabled, USE\_TICKLESS\_IDLE set to Disabled, USE\_TASK\_NOTIFICATIONS set to Enabled, RECORD\_STACK\_HIGH\_ADDRESS set to Disabled), 'Memory management settings' (Memory Allocation set to Dynamic / Static, TOTAL\_HEAP\_SIZE set to 40000 Bytes, Memory Management Scheme set to heap\_4), and 'Hook function related definitions' (USE\_IDLE\_HOOK set to Disabled, USE\_TICK\_HOOK set to Disabled).

Parameter	Value
FreeRTOS version	10.3.1
CMSIS-RTOS version	2.00
ENABLE_MPUSet	Disabled
ENABLE_FPU	Disabled
USE_PREEMPTION	Enabled
CPU_CLOCK_HZ	SystemCoreClock
TICK_RATE_HZ	1000
MAX_PRIORITIES	56
MINIMAL_STACK_SIZE	128 Words
MAX_TASK_NAME_LEN	16
USE_16_BIT_TICKS	Disabled
IDLE_SHOULD_YIELD	Enabled
USE_MUTEXES	Enabled
USE_RECURSIVE_MUTEXES	Enabled
USE_COUNTING_SEMAPHORES	Enabled
QUEUE_REGISTRY_SIZE	8
USE_APPLICATION_TASK_TAG	Disabled
ENABLE_BACKWARD_COMPATIBILITY	Enabled
USE_PORT_OPTIMISED_TASK_SELECTION	Disabled
USE_TICKLESS_IDLE	Disabled
USE_TASK_NOTIFICATIONS	Enabled
RECORD_STACK_HIGH_ADDRESS	Disabled
Memory Allocation	Dynamic / Static
TOTAL_HEAP_SIZE	40000 Bytes
Memory Management Scheme	heap_4
USE_IDLE_HOOK	Disabled
USE_TICK_HOOK	Disabled

# Middleware & Software Packs

## FreeRTOS

The screenshot shows the STcube software interface with the following details:

- Pinout & Configuration** tab is active.
- Clock Configuration** tab is visible.
- Project Manager** tab is visible.
- Software Packs** section is selected under Clock Configuration.
- Interface**: CMSIS\_V2
- Mode**: FREERTOS Mode and Configuration
- Configuration** tab is selected.
- Reset Configuration** button is present.
- Parameter Selection Buttons**: Config parameters, Include parameters, Advanced settings (highlighted), User Constants, Tasks and Queues, Timers and Semaphores, Mutexes, Events, FreeRTOS Heap Usage.
- Configure the below parameters :**

  - Newlib settings (see parameter description first)**: USE\_NEWLIB\_REENTRANT (Enabled)
  - Project settings (see parameter description first)**: Use FW pack heap file

A pink arrow points from the text "Enable USE\_NEWLIB\_REENTRANT" at the bottom right to the "Enabled" status of the USE\_NEWLIB\_REENTRANT parameter.

Enable  
USE\_NEWLIB\_REENTRANT



# NVIC

Categories A-Z

System Core

- ✓ DMA
- ✓ GPIO
- IWDG
- ✓ NVIC**
- ✓ RCC
- ⚠ SYS
- WWDG

Analog >

Timers >

Connectivity >

Multimedia >

Computing >

Middleware and Software Packs >

Utilities >

NVIC Mode and Configuration

Mode

Configuration

✓ NVIC    ✓ Code generation

Sort by Preemption Priority and Sub Priority     Sort by interrupts names

Search  Show available interrupts  Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Uses FreeRTOS functions
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"/>
System tick timer	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
PVD through EXTI line 16, PVM (monit. VDDIO2) thro...	<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"/>
EXTI line 2 and line 3 interrupts	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
EXTI line 4 to 15 interrupts	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
DMA1 channel 1 interrupt	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
DMA1 channel 2 and channel 3 interrupts	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
Time base: TIM1 break, update, trigger and commuta...	<input checked="" type="checkbox"/>	3	<input type="checkbox"/>
SPI1/I2S1 Interrupt	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>
USART2 + LPUART2 Interrupt	<input checked="" type="checkbox"/>	3	<input checked="" type="checkbox"/>

# Middleware & Software Packs

## X-CUBE-ST67W61 Software package

Pack / Bundle / Component	Status	Version	Selection
STMicroelectronics.X-CUBE-ST67W61	✓ 1.0.0		
Device Applications			
Application	✓ 1.0.0		BLE_Commissioning_v
Freertos_Tickless		1.0.0	<input type="checkbox"/>
Network ST67W6X_Network_Driver	✓ 1.0.0		
ServiceShell		1.0.0	<input type="checkbox"/>
ServiceAPI	✓ 1.0.0		<input checked="" type="checkbox"/>
Driver / W61_at_and_bus	✓ 1.0.0		<input checked="" type="checkbox"/>
Utilities	✓		
Logging	✓ 1.0.0		<input checked="" type="checkbox"/>
Shell		1.0.0	<input type="checkbox"/>
Statistics		1.0.0	<input type="checkbox"/>
Debug TraceRecorder		4.10.2	
TraceRecorder		4.10.2	<input type="checkbox"/>
Data Exchange cJSON		1.7.18	
cJSON		1.7.18	<input type="checkbox"/>
File System LittleFS		2.10.1	
LittleFS		2.10.1	<input type="checkbox"/>
Utility Utilities		1.4.2	
LPM / Tiny LPM		1.4.2	<input type="checkbox"/>

Select BLE\_Commissioning\_v

Enable ServiceAPI

Enable Driver/W61\_at\_and\_bus

Enable Logging

# Middleware & Software Packs

## X-CUBE-ST67W61 Software package

The screenshot shows the X-CUBE-ST67W61 software configuration interface. The left panel, titled "Pinout & Configuration", displays a list of categories under "X-CUBE-ST67W61". The "X-CUBE-ST67W61" item is selected and highlighted with a pink border. The right panel, titled "Clock Configuration", shows the "Mode" section with checkboxes for "Device Applications" and "Network ST67W6X Network Driver", both of which are checked. Below this is the "Configuration" section with tabs for "Reset Configuration", "W6X Modules", "Parameter Settings", "User Constants", and "Platform Settings", where "Platform settings" is highlighted with a pink arrow. The "BSP" section lists components and their found solutions:

Name	IPs or Components	Found Solutions	BSP API
LogSh_UART	USART:Asynchronous	USART2	Unknown
NCP_SPI	SPI:Full-Duplex Master	SPI1	Unknown



# Middleware & Software Packs

## X-CUBE-ST67W61 Software package

The screenshot shows the X-CUBE-ST67W61 software configuration interface. On the left, the 'Pinout & Configuration' tab is active, displaying a tree view of software packages. The 'X-CUBE-ST67W61' package is selected and highlighted with a blue bar at the bottom. On the right, the 'Clock Configuration' tab is active, showing the 'STMicroelectronics X-CUBE-ST67W61.1.0.0 Mode and Configuration' screen. It includes sections for 'Mode' (Device Applications, Network ST67W6X Network Driver), 'Configuration' (Reset Configuration, W6X Modules, Parameter Settings, User Constants, Platform Settings), and a detailed parameter list. A callout box with a pink arrow points to the 'W6X\_BLE\_HOSTNAME' parameter, with the text: 'Rename W6X\_BLE\_HOSTNAME to something unique'. Another callout box with a pink arrow points to the 'W6X\_POWER\_SAVE\_AUTO' parameter, with the text: 'Disable W6X\_POWER\_SAVE\_AUTO'.

Pinout & Configuration

Clock Configuration

STMicroelectronics X-CUBE-ST67W61.1.0.0 Mode and Configuration

Mode

Device Applications

Network ST67W6X Network Driver

Configuration

Reset Configuration

W6X Modules Parameter Settings User Constants Platform Settings

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

- LOG\_OUTPUT\_MODE
- FREERTOS\_TASK\_NOTIF\_ARRAY\_LEN

Logging-Shell

- LOG\_LEVEL

W6X Parameters

- W6X\_BLE\_HOSTNAME
- W6X\_WIFI\_HOSTNAME
- W6X\_POWER\_SAVE\_AUTO
- W6X\_CLOCK\_MODE
- W6X\_WIFI\_AUTOCONNECT
- W6X\_WIFI\_DHCP\_MODE
- W6X\_WIFI\_COUNTRY\_CODE
- W6X\_WIFI\_ADAPTIVE\_COUNTRY\_CODE
- W6X\_NET\_RECV\_TIMEOUT (ms)
- W6X\_NET\_SEND\_TIMEOUT (ms)
- W6X\_NET\_RECV\_BUFFER\_SIZE
- W6X\_WIFI\_DNS\_MANUAL
- W6X\_WIFI\_DNS\_IP\_1
- W6X\_WIFI\_DNS\_IP\_2
- W6X\_WIFI\_DNS\_IP\_3

W61 drv Parameters

- W61\_WIFI\_MAX\_DETECTED\_AP
- W61\_BLE\_MAX\_CONN\_NBR
- W61\_BLE\_MAX\_DETECTED\_PERIPHERAL
- W61\_BLE\_MAX\_SERVICE\_NBR

LOG\_OUTPUT\_UART  
8

LOG\_DEBUG  
ST67W61\_BLE  
ST67W61\_WIFI  
No

Internal RC oscillator  
No  
DHCP STA+SAP  
00  
No  
5000  
5000  
9216  
No  
{208, 67, 222, 222}  
{8, 8, 8, 8}  
{0, 0, 0, 0}

50  
1  
10  
5

Utilities >

Rename W6X\_BLE\_HOSTNAME to something unique

Disable W6X\_POWER\_SAVE\_AUTO

ST Restricted

# Project Creation

The screenshot shows the STM32CubeMX Project Manager interface for a project named "ST67\_G0\_BLE\_Comm". The interface is divided into several sections: Pinout & Configuration, Clock Configuration, Project Manager, Project (selected), Code Generator, and Advanced Settings. In the Project Manager section, there is a checkbox labeled "Generate Under Root" which is checked. In the Advanced Settings section, there is a checkbox labeled "Use Default Firmware Location" which is also checked. A pink arrow points from the text "1. Enable ‘Generate Under Root’" to the "Generate Under Root" checkbox. Another pink arrow points from the text "2. Enable ‘Use Default Firmware Location’" to the "Use Default Firmware Location" checkbox.

Here, we are enabling the Flat directory structure: (i.e. Driver and MW directories copied into your project directory)



2. Enable “Use Default Firmware Location”

# Code Generator

Home > STM32G0B1RETx - NUCLEO-G0B1RE > ST67\_G0\_BLE\_Comm.ioc - Project Manager >

Pinout & Configuration | Clock Configuration | Project Manager

**Project**

STM32Cube MCU packages and embedded software packs

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

**Code Generator** (highlighted with a red box)

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

Advanced Settings

HAL Settings

Set all free pins as analog (to optimize the power consumption)  
 Enable Full Assert

**3. Enable “Copy only the necessary files”**

# Advanced settings

Code Generator

> SPI  
> USART  
STMicroelectronics.X-CUBE-ST67W61.1.0.0

HAL  
HAL  
HAL

Advanced Settings

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance Name	<input type="checkbox"/> Do Not Generate Function Call	<input type="checkbox"/> Visibility (Static)
<input checked="" type="checkbox"/>	1	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	2	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	3	MX_DMA_Init	DMA	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	4	MX_SPI1_Init	SPI1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	5	MX_USART2_UART_Init	USART2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	6	MX_App_BLE_Commis...	STMicroelectronics.X-C...	<input checked="" type="checkbox"/>	<input type="checkbox"/>

5. Uncheck this option for SPI1 and USART2

6. Check Visibility for SPI1 and USART2

4. Enable this option (here: we are not generating this function call)

# Generate Code

## 5. Generate Code

The screenshot shows the STMicroelectronics Project Manager interface for an STM32G0B1RETx - NUCLEO-G0B1RE project named G0\_Experts\_Training.ioc. The interface is divided into several tabs: Pinout & Configuration, Clock Configuration (selected), Project Manager, and Tools.

**Project** section:

- Driver Selector: Search (Ctrl+F)
- RCC: HAL
- GPIO: HAL
- DMA: HAL
- > SPI: HAL
- > USART: HAL
- STMicroelectronics.X-CUBE-ST67W61.1.0.0

**Code Generator** section:

- HAL

**Advanced Settings** section:

Generated Function Calls:

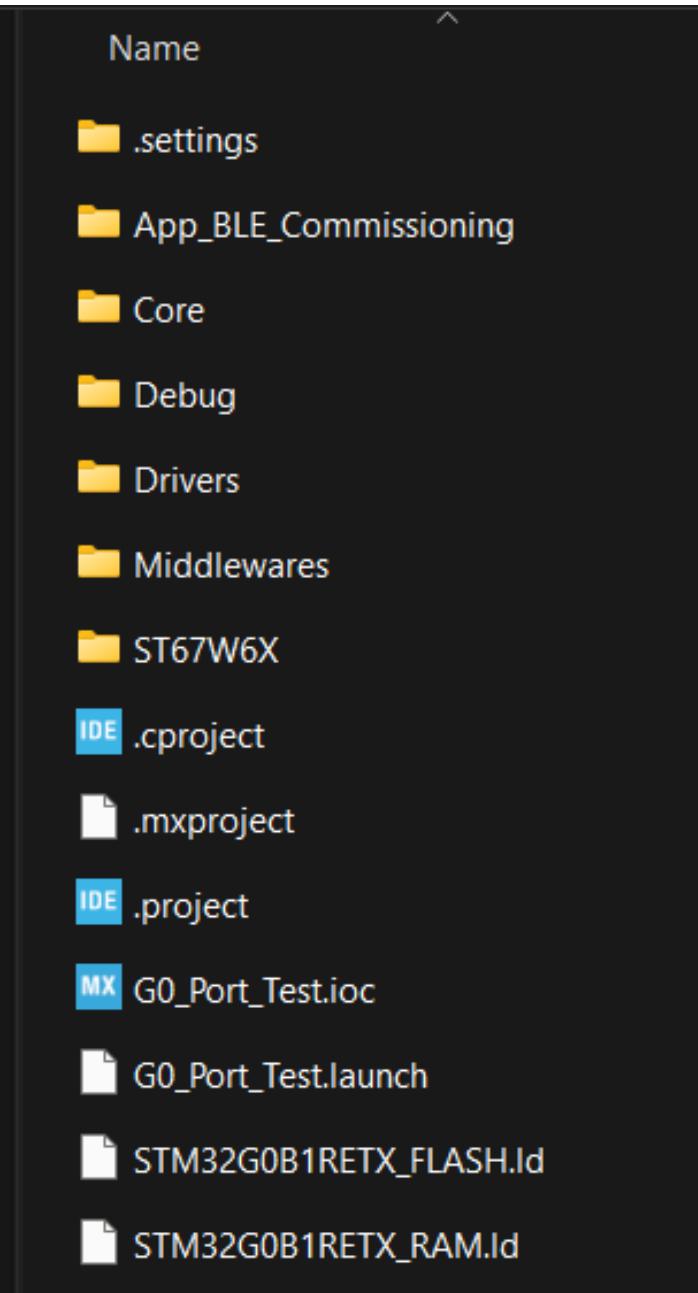
Generate Code	Rank	Function Name	Peripheral Instance Name	<input type="checkbox"/> Do Not Generate Function Call	<input checked="" type="checkbox"/> Visibility (Static)
<input checked="" type="checkbox"/>	1	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	2	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	3	MX_DMA_Init	DMA	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	4	MX_SPI1_Init	SPI1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	5	MX_USART2_UART_Init	USART2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	6	MX_App_BLE_Commiss...	STMicroelectronics.X-C...	<input checked="" type="checkbox"/>	<input type="checkbox"/>

**Tools** section:

Register CallBack:

Search (Ctrl+F)	ADC	DISABLE
Q	CEC	DISABLE
COMP	COMP	DISABLE
CRYP	CRYP	DISABLE
DAC	DAC	DISABLE
FDCAN	FDCAN	DISABLE
HCD	HCD	DISABLE
I2C	I2C	DISABLE
I2S	I2S	DISABLE
IRDA	IRDA	DISABLE
LPTIM	LPTIM	DISABLE
PCD	PCD	DISABLE
RNG	RNG	DISABLE
RTC	RTC	DISABLE
SMBUS	SMBUS	DISABLE
SPI	SPI	DISABLE
TIM	TIM	DISABLE
UART	UART	DISABLE
USART	USART	DISABLE
WWDG	WWDG	DISABLE

# Folder Structure



Folder structure view after generating the code from CubeMX

- App code
- Core
- Drivers
- Middlewares
- ST67W6X

# STM32CubeIDE

Load the project on the STM32CubeIDE

The screenshot shows the STM32CubeIDE interface. On the left, the Project Explorer displays the project structure for 'ST67\_G0\_BLE\_Comm'. A red arrow points to the project name 'ST67\_G0\_BLE\_Comm'. The main window shows the 'main.c' file with the following code:

```
411 /* USER CODE END 4 */  
412  
413 /* USER CODE BEGIN Header_StartDefaultTask */  
414 /**  
415  * @brief  Function implementing the default task  
416  * @param  argument: Not used  
417  * @retval None  
418 */  
419  
420 /* USER CODE END Header_StartDefaultTask */  
421 void StartDefaultTask(void *argument)  
422 {  
423     /* USER CODE BEGIN 5 */  
424     main_app();  
425     /* Infinite loop */  
426     for(;;)  
427     {  
428         osDelay(1000);  
429     }  
430     /* USER CODE END 5 */  
431 }  
432  
433 /**  
434  * @brief  Period elapsed callback in non-isolated mode  
435  * @note   This function is called when a direct call is made  
436  *        to a TIM PeriodElapsedCallback function from a timer  
437  *        interrupt.  
438  *        It makes a direct call to the user supplied function.  
439  * @param  htim : TIM handle  
440  * @retval None  
441 */  
442 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)  
{  
    /* If no LICENSE file comes with this software, you can include the license terms directly here. */  
    /* ****endl**** */  
}  
/* USER CODE END Header */  
/* Includes -----*/  
#include "main.h"  
#include "cmsis_os.h"  
  
/* Private includes -----*/  
/* USER CODE BEGIN Includes */  
#include "main_app.h"  
/* USER CODE END Includes */
```

2. Call #include "main\_app.h"

1. Call main\_app() function under StartDefaultTask() in main.c file

The screenshot shows the 'main.h' header file with the following code:

```
14 * If no LICENSE file comes with this software, you can include the license terms directly here.  
15 * ****endl****  
16 * ****endl****  
17 */  
18 /* USER CODE END Header */  
19 /* Includes -----*/  
20 #include "main.h"  
21 #include "cmsis_os.h"  
22  
23 /* Private includes -----*/  
24 /* USER CODE BEGIN Includes */  
25 #include "main_app.h"  
26 /* USER CODE END Includes */
```



# xPortIsInsideInterrupt()

- If the used device embeds an Arm® Cortex®M0+ (e.g. STM32G0) and its FreeRTOS version (v10.3.1) which is older than V10.6, definition below **must be added manually after project has been generated.**
- The below function definition must be added in the **portmacro.h** file (Project/Middlewares/Third\_Party/FreeRTOS/Source/portable/GCC/ARM\_CM0)

```
#define portINLINE __inline
#ifndef portFORCE_INLINE
#define portFORCE_INLINE inline __attribute__((always_inline))
#endif

/*-----------------------------------------------------*/
portFORCE_INLINE static BaseType_t xPortIsInsideInterrupt( void )
{
    uint32_t ulCurrentInterrupt;
    BaseType_t xReturn;

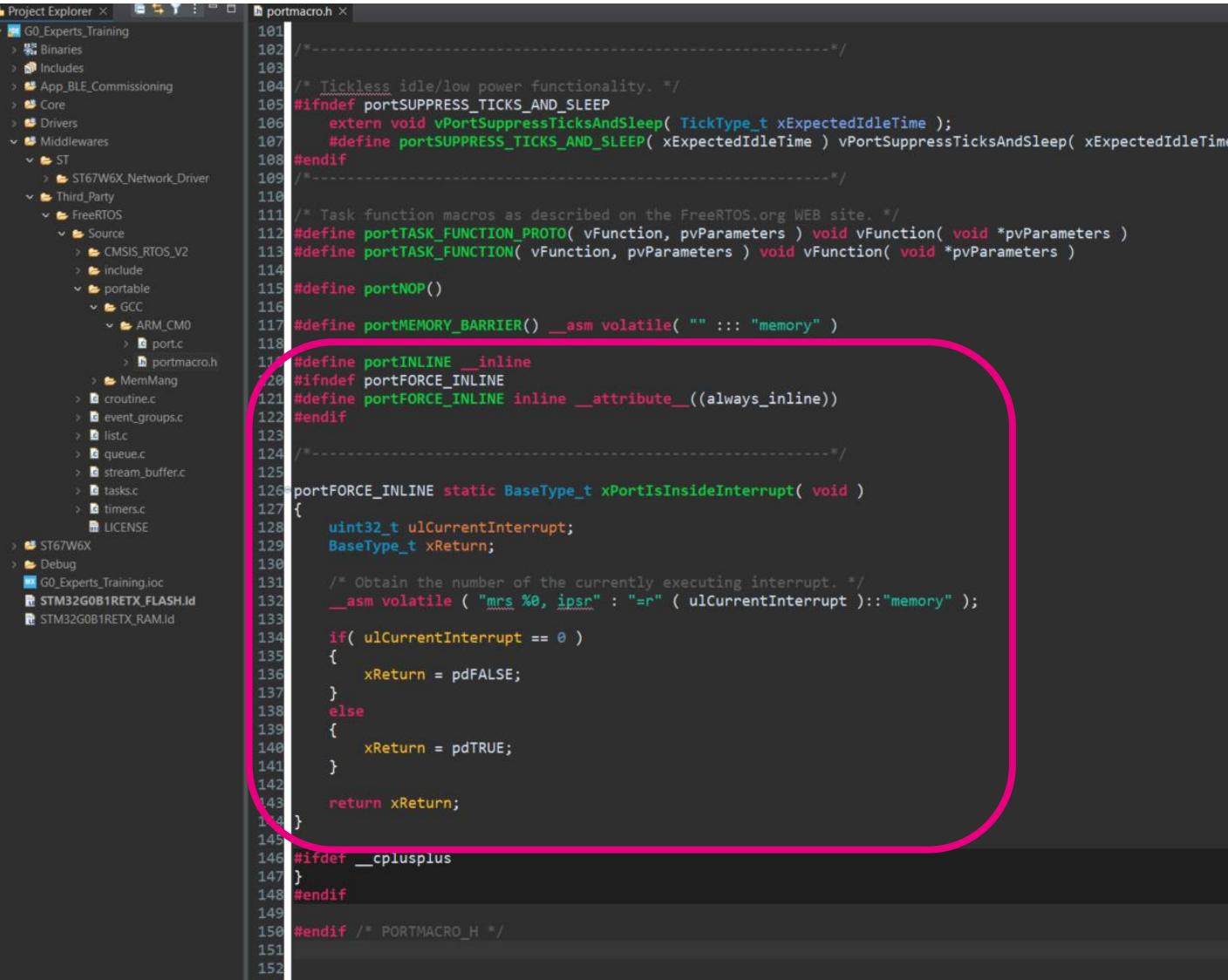
    /* Obtain the number of the currently executing interrupt.*/
    __asm volatile ( "mrs %0, ipsr" : "=r" ( ulCurrentInterrupt )::"memory" );

    if( ulCurrentInterrupt == 0 )
    {
        xReturn = pdFALSE;
    }
    else
    {
        xReturn = pdTRUE;
    }

    return xReturn;
}
```



# STM32CubeIDE



```
Project Explorer X portmacro.h X
G0_Experts_Training
  Binaries
  Includes
  App_BLE_Commissioning
  Core
  Drivers
  Middlewares
    ST
      ST67W6X_Network_Driver
  Third_Party
    FreeRTOS
      Source
        CMSIS_RTOS_V2
      include
      portable
        GCC
          ARM_CM0
            port.c
            portmacro.h
        MemMang
        routine.c
        event_groups.c
        list.c
        queue.c
        stream_buffer.c
        tasks.c
        timers.c
        LICENSE
    ST67W6X
    Debug
    G0_Experts_Training.ioc
    STM32G0B1RETX_FLASH.Id
    STM32G0B1RETX_RAM.Id

101  /*
102   *-----*
103   /* Tickless idle/low power functionality. */
104   #ifndef portSUPPRESS_TICKS_AND_SLEEP
105     extern void vPortSuppressTicksAndSleep( TickType_t xExpectedIdleTime );
106     #define portSUPPRESS_TICKS_AND_SLEEP( xExpectedIdleTime ) vPortSuppressTicksAndSleep( xExpectedIdleTime )
107   #endif
108   /*-----*/
109
110  /* Task function macros as described on the FreeRTOS.org WEB site. */
111  #define portTASK_FUNCTION_PROTO( vFunction, pvParameters ) void vFunction( void *pvParameters )
112  #define portTASK_FUNCTION( vFunction, pvParameters ) void vFunction( void *pvParameters )
113
114  #define portNOP()
115
116  #define portMEMORY_BARRIER() __asm volatile( "" :::"memory" )
117
118  #define portINLINE __inline
119  #ifndef portFORCE_INLINE
120    #define portFORCE_INLINE inline __attribute__((always_inline))
121  #endif
122
123  /*-----*/
124
125  portFORCE_INLINE static BaseType_t xPortIsInsideInterrupt( void )
126  {
127    uint32_t ulCurrentInterrupt;
128    BaseType_t xReturn;
129
130    /* Obtain the number of the currently executing interrupt. */
131    __asm volatile ( "mrs %0, ipsr" : "=r" ( ulCurrentInterrupt ):::"memory" );
132
133    if( ulCurrentInterrupt == 0 )
134    {
135      xReturn = pdFALSE;
136    }
137    else
138    {
139      xReturn = pdTRUE;
140    }
141
142    return xReturn;
143  }
144
145  #ifdef __cplusplus
146  }
147  #endif
148
149
150  #endif /* PORTMACRO_H */
151
152
153
```

Call xPortIsInsideInterrupt() function under portmacro.h file

IMP: As we are using G0 (CortexM0) -> older version of FreeRTOS, so we need to add this function definition manually.

```
#define portINLINE __inline
#ifndef portFORCE_INLINE
#define portFORCE_INLINE inline
__attribute__((always_inline))
#endif

/*-----*/

portFORCE_INLINE static BaseType_t xPortIsInsideInterrupt(
void )
{
  uint32_t ulCurrentInterrupt;
  BaseType_t xReturn;

  /* Obtain the number of the currently executing interrupt. */
  __asm volatile ( "mrs %0, ipsr" : "=r" ( ulCurrentInterrupt
):::"memory" );

  if( ulCurrentInterrupt == 0 )
  {
    xReturn = pdFALSE;
  }
  else
  {
    xReturn = pdTRUE;
  }

  return xReturn;
}
```

# STM32CubeIDE

Add `#ifndef STM32G0B1xx`

The screenshot shows the STM32CubeIDE interface. On the left is the Project Explorer, displaying a project structure for ST67\_G0\_BLE\_Comm. The code editor on the right shows the file spi\_port.c. A red box highlights the line `#ifndef STM32G0B1xx`. The code implements a custom memcpy function for the ST67W6X chip, handling byte-aligned and unaligned memory copy operations.

```
76 /* USER CODE END PFP */
77
78 /* Functions Definition */
79 #ifndef STM32G0B1xx
80 #ifdef __ICCARM__
81 void *spi_port_memcpy(void *dest, const void *src, unsigned int len)
82#endif /* __ICCARM__ */
83 #ifdef __GNUC__
84 void *memcpy(void *dest, const void *src, unsigned int len)
85#endif /* __ICCARM__ */
86 {
87     /* USER CODE BEGIN memcpy_1 */
88
89     /* USER CODE END memcpy_1 */
90     uint8_t *d = (uint8_t *)dest;
91     const uint8_t *s = (const uint8_t *)src;
92
93     /* Copy bytes until the destination address is aligned to 4 bytes */
94     while (((uint32_t) d % 4 != 0) && len > 0)
95     {
96         *d++ = *s++;
97         len--;
98     }
99
100    /* Copy 4-byte blocks */
101    uint32_t *d32 = (uint32_t *)d;
102    const uint32_t *s32 = (const uint32_t *)s;
103    while (len >= 4)
104    {
105        *d32++ = *s32++;
106        len -= 4;
107    }
108
109    /* Copy remaining bytes */
110    d = (uint8_t *)d32;
111    s = (const uint8_t *)s32;
112    while (len > 0)
113    {
114        *d++ = *s++;
115        len--;
116    }
117
118    return dest;
119    /* USER CODE BEGIN memcpy_End */
120
121    /* USER CODE END memcpy_End */
122}
123#endif
```

Here we are disabling the custom `memcpy()` definition, as we would be using the `memcpy()` definition in `string.h`

Project/ST67W6X/spi\_port.c

Add `#endif`



# Build!

Build the project

The screenshot shows the STM32CubeIDE interface with the following details:

- Project Explorer:** Shows the project structure under "ST67\_G0\_BLE\_Comm".
- Code Editor:** Displays the file "portmacro.h" with code related to FreeRTOS porting.
- Build Status:** The bottom console window shows the build command:

```
arm-none-eabi-gcc -o "ST67_G0_BLE_Comm.elf" @"objects.list" -mcpu=cortex-m0plus -T"C:\ST67_Test\ST67_G0_BLE_Comm\linker\linker.ld"
```

and the output:

```
Finished building target: ST67_G0_BLE_Comm.elf

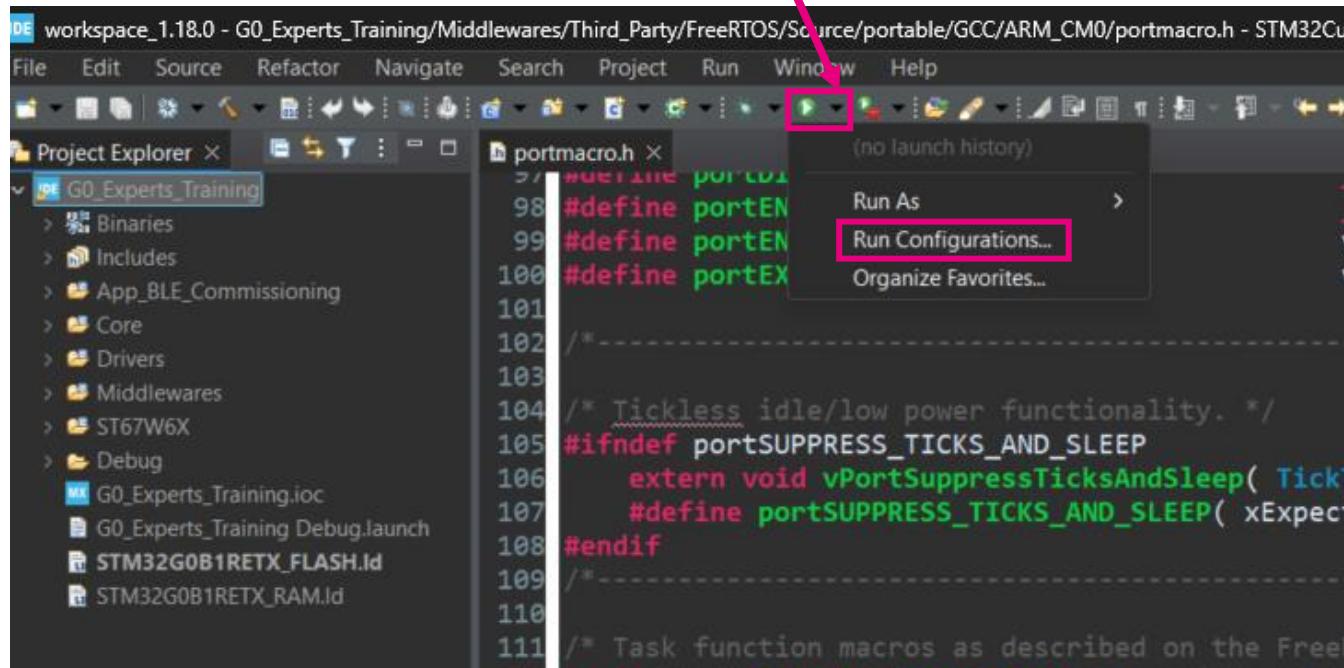
arm-none-eabi-size ST67_G0_BLE_Comm.elf
arm-none-eabi-objdump -h -S ST67_G0_BLE_Comm.elf > "ST67_G0_BLE_Comm.list"
    text      data      bss      dec filename
 126308       328     47944   174580  2a9f4 ST67_G0_BLE_Comm.elf
Finished building: default.size.stdout

Finished building: ST67_G0_BLE_Comm.list
```
- Build Progress:** A pink arrow points from the "Build the project" button to the "Build" icon in the toolbar.

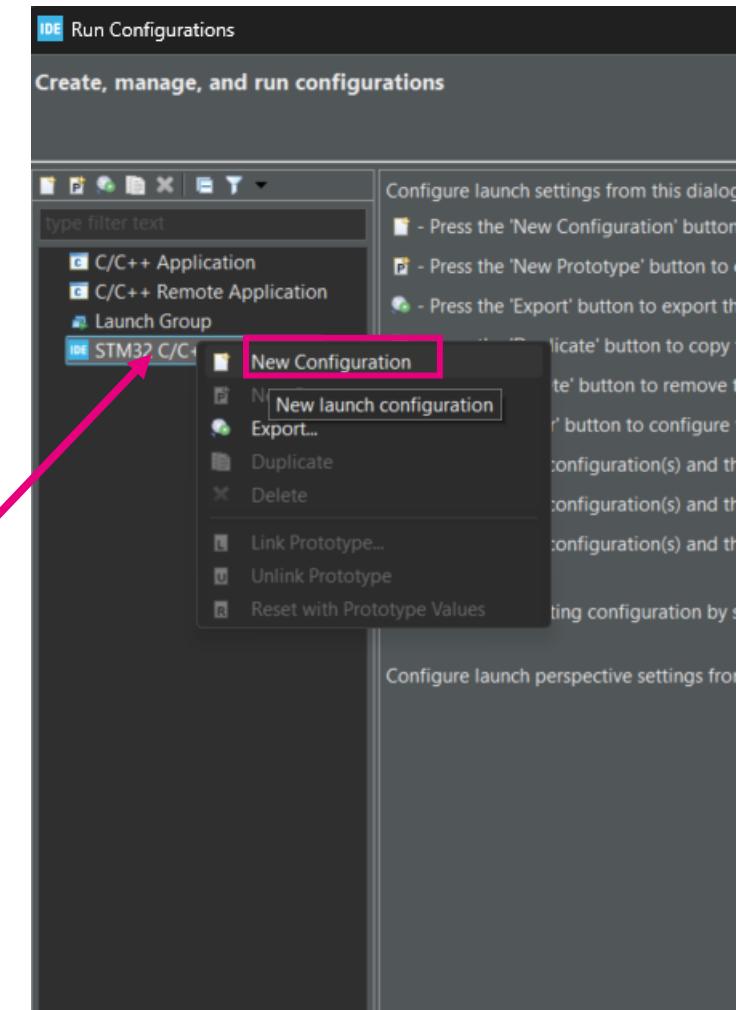


# Flash!

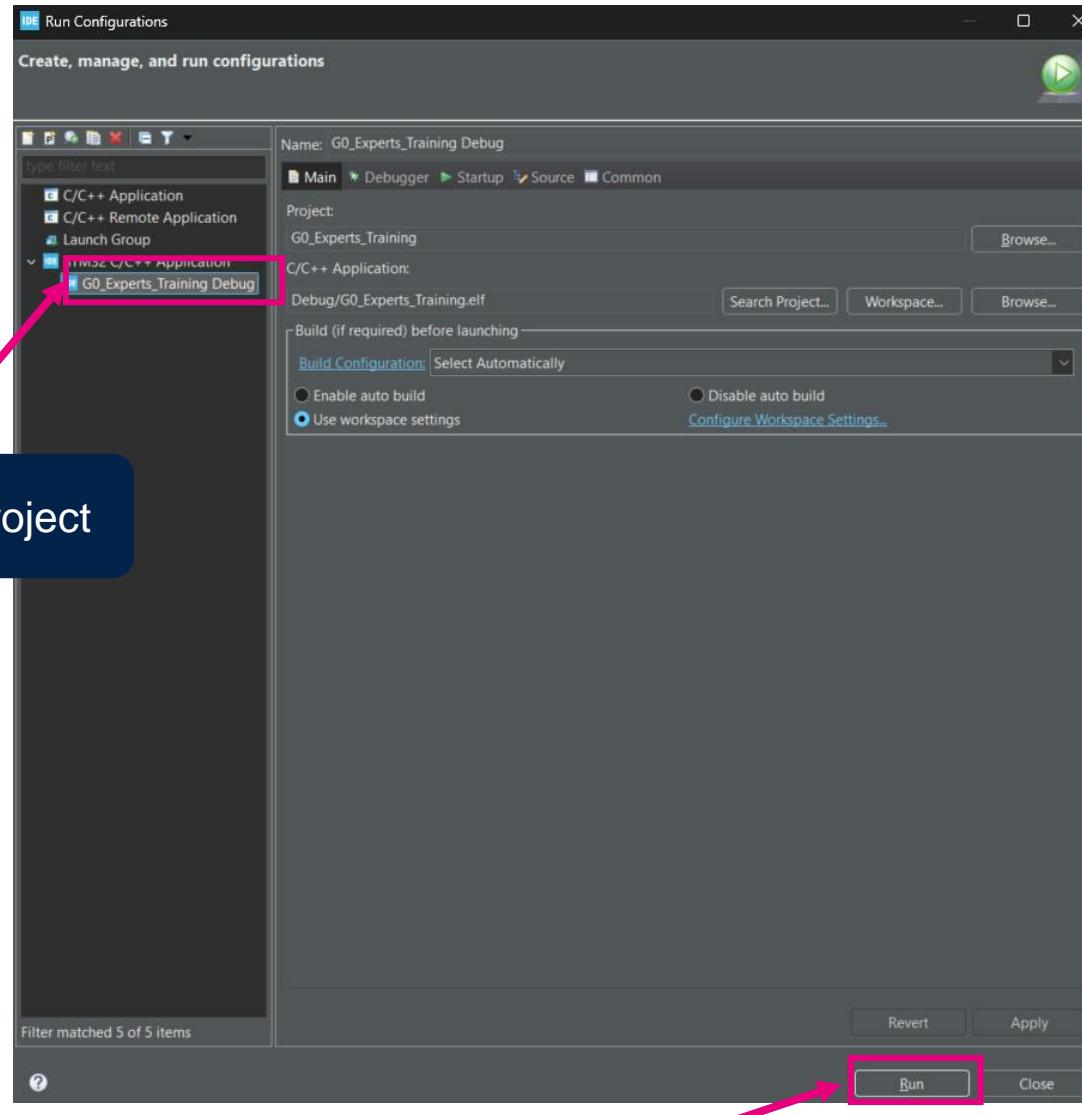
Flash the project



Right-Click on STM32  
C/C++ application



# Flash!



Console logs  
after flashing  
the project

The screenshot shows the STM32CubeProgrammer console window. A pink arrow points from a blue button labeled 'Console logs after flashing the project' to the top of the console window. The console displays log output from the ST-LINK GDB server, detailing the flash process. It shows the device being identified as an STM32G0B0xx/B1xx/C1xx, the memory being programmed, the file being downloaded, and finally the download being verified successfully.

```
terminated> G0_Experts_Training Debug [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Aug 20, 2025, 3:14:43PM) [pid: 113128]

Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...

STM32CubeProgrammer v2.19.0

Log output file: C:\Users\kulkarni\AppData\Local\Temp\STM32CubeProgrammer_a15436.log
ST-LINK SN : 0670F485570854967113128
ST-LINK FW : V2J46M31
Board : NUCLEO-G0B1RE
Voltage : 3.23V
SWD freq : 4000 KHz
Connect mode: Under Reset
Reset mode : Hardware reset
Device ID : 0x467
Revision ID : Rev Z
Device name : STM32G0B0xx/B1xx/C1xx
Flash size : 512 KBytes
Device type : MCU
Device CPU : Cortex-M0+
BL Version : 0x92

Opening and parsing file: ST-LINK_GDB_server_a15436.srec

Memory Programming ...
File : ST-LINK_GDB_server_a15436.srec
Size : 122.02 KB
Address : 0x08000000

Erasing memory corresponding to sector 0:
Erasing internal memory sectors [0 61]
Download in Progress:

File download complete
Time elapsed during download operation: 00:00:03.410

Verifying ...

Download verified successfully

Shutting down...
Exit.
```



# Tera Term Logs

**Tera Term Logs after flashing**

The screenshot shows the Tera Term interface with two windows. The main window displays the logs from the ST67W6X Wi-Fi Commissioning over BLE Application. The logs include host information, battery voltage, MAC address, and BLE configuration details. A pink arrow points from the text "Tera Term Logs after flashing" to the start of the log output. The second window is a "Serial port setup and connection" dialog, which is currently set to Port: COM21, Speed: 115200, Data: 8 bit, Parity: none, Stop bits: 1 bit, and Flow control: none. It also includes fields for Transmit delay (0 msec/char and 0 msec/line) and a list of device details at the bottom.

```
##### Welcome to ST67W6X Wi-Fi Commissioning over BLE Application #####
# build: 12:23:03 Jul 28 2025
----- Host info -----
Host FW Version: 1.0.0
----- ST67W6X info -----
ST67W6X MW Version: 1.0.0
AT Version: 1.0.0.1
SDK Version: 2.0.75
MAC Version: 1.6.38
Build Date: May 10 2025 10:15:04
Module ID:
BOM ID: 0
Manufacturing Year: 2000
Manufacturing Week: 00
Battery Voltage: 3.316 V
Trim Wi-Fi hp: 8.8,8.8,8.7,6,6,7,7,7,8,8,8
Trim Wi-Fi lp: 8.8,8.8,9,9,9,9,10,10,10,11,11,11
Trim BLE: 7,5,5,7,7
Trim XTAL: 32
MAC Address: 40:82:7b:00:0c:2d
Anti-rollback Bootloader: 0
Anti-rollback App: 0

Wi-Fi init is done
Net init is done
Ble init is done
BD Address: 40:82:7b:00:0c:2e
Configure BLE
BLE configuration is done

BLE Commissioning Service Creation
- BLE service created
- BLE WIFI Control charac created
- BLE WIFI Configure charac created
- BLE WIFI Monitoring charac created
- BLE services and charac registered
BLE service and charac creation is done

Start BLE advertising
BLE advertising is started

```

Tera Term: Serial port setup and connection

Port:	COM21	New setting
Speed:	115200	Cancel
Data:	8 bit	Help
Parity:	none	
Stop bits:	1 bit	
Flow control:	none	

Transmit delay

Device Friendly Name: STMicroelectronics STLink Virtual COM P  
Device Instance ID: USBVID\_0483&PID\_374B&MI\_02&6&38AF34B  
Device Manufacturer: STMicroelectronics  
Provider Name: STMicroelectronics  
Driver Date: 4-1-2021  
Driver Version: 2.2.0.0





# Documentation





## Access:

The wifi wiki is opened to all



## Wiki address of the main page

[https://wiki.st.com/stm32mcu/wiki/Connectivity:  
Introduction\\_to\\_Wi-Fi](https://wiki.st.com/stm32mcu/wiki/Connectivity:Introduction_to_Wi-Fi)



We will be happy to get your  
feedback!

# Wiki Wi-Fi®: pages breakdown

About Wi-Fi	1 Wi-Fi® overview 1.1 What is Wi-Fi® 1.2 Wi-Fi® network architecture 1.3 Wi-Fi® features details
Details about ST67W611M1	2 ST67W611M1 2.1 ST67W611M1 overview 2.2 X-CUBE-ST67W61 2.3 ST67W611M1: Hardware and board description 2.4 ST67W611M1 Security overview 2.5 ST67W611M1-based customer end-Product 2.6 ST67W611M1 evaluation
X-CUBE-ST67W61	3 X-CUBE-ST67W61 software application notes and user manuals 3.1 Getting started with ST67W611M1
Host boards supported	4 ST67W611M1 associated hosts boards
Tools & References	5 Specific tools 6 Terms and definitions 7 References

# Other documentation

## 2

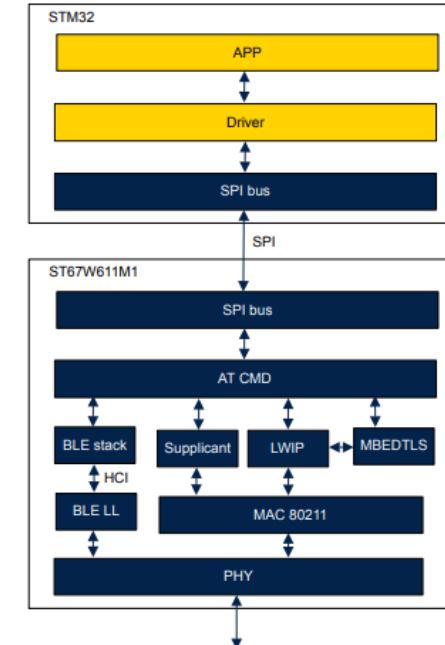
### X-CUBE-ST67W61 architecture overview

- Getting started with X-CUBE-ST67W61 [here](#)

The X-CUBE-ST67W61 is a set of software components implementing host applications driving a Wi-Fi® and Bluetooth® LE coprocessor. The coprocessor is controlled via AT command over SPI interface.

The figure below illustrates the architecture of the solution:

Figure 1. X-CUBE-ST67W61 architecture view



- MW API documentation is available:
  - X-CUBE-ST67W61\_V1.0.0\Middlewares\ST\ST67W6X\_Network\_Driver\Doc\ST67W6X\_Network\_Driver.chm

# ST Customer Support

- [ST Support Home](#)
  - Tickets can be submitted under the part number X-CUBE-ST67W61 or ST67W61M1



# Our technology starts with You



Find out more at [www.st.com](http://www.st.com)

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to [www.st.com/trademarks](http://www.st.com/trademarks).

All other product or service names are the property of their respective owners.

