STM32 connectivity expansion pack

User guide

About this document

Version

1.2.0

Scope and purpose

The STM32 Connectivity Expansion Pack is an extension of the CMSIS-Pack standard established by Arm. The pack is compliant with the full CMSIS-Pack standard, with additional requirements/restrictions on the final pack to meet the STM standard. This pack uses libraries from the ModusToolbox environment. For more details, refer to https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software You can select and configure the pack in the STM32CubeMX tool, make choices appropriate for your design, such as which CYW43xxx device to use, and then generate a project from your selection.

Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, menus and sub-menus
Italics	Denotes file names and paths.
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
File > New	Indicates that a cascading sub-menu opens when you select a menu item

Abbreviations and definitions

The following define the abbreviations and terms used in this document:

BSP – Board Support Package

PAL – Platform Adaptation Layer

WCM – Wi-Fi Connection Manager

WHD – Wi-Fi Host Driver

Table of contents

Table of contents

1 Expansion pack contents

The following table shows the components and their versions included with the expansion pack:

Component name	Version	Details
abstraction-rtos	1.5.0	The RTOS abstraction layer provides simple RTOS services like threads, semaphores, mutexes, queues, and timers. It is not intended to be a full features RTOS interface, but they provide just enough support to allow for RTOS independent drivers and middleware.
bluetooth-freertos	3.4	WICED Bluetooth host stack solution includes bluetooth stack library, bluetooth controller firmware and platform/os porting layer. This component is compatible with Theadx as well.
btstack	3.5	BTSTACK is Infineon's Bluetooth Host Protocol Stack implementation. The stack is optimized to work with Infineon Bluetooth controllers. The BTSTACK supports Bluetooth BR/EDR and BLE core protocols.
connectivity-utilities	4.0.0	The connectivity utilities library is a collection of general-purpose middleware utilities such as: JSON parser, Linked list, String utilities, Network helpers, Logging functions, and Middleware Error codes. Several connectivity middleware libraries will depend on this library.
core-lib	1.2.0	The Core Library provides basic types and utilities that can be used between different devices. This allows different libraries to share common items between themselves to avoid reimplementation and promote consistency.
device	1.2.0	Selects appropriate CYW43xxx firmware and drivers for selected connectivity device.
lwIP	2.1.2	lwIP is a small independent implementation of the TCP/IP protocol suite. The focus of the lwIP TCP/IP implementation is to reduce the RAM usage while still having a full-scale TCP. This making lwIP suitable for use in embedded systems with tens of kilobytes of free RAM and room for around 40 kilobytes of code ROM.
pal	1.2.0	Infineon-STM32 Platform Adaptation Layer (PAL).
wifi-host-driver	2.4.0	The Wi-Fi host driver (WHD) is an independent, embedded driver that provides a set of APIs to interact with Infineon WLAN chips. The WHD is an independent firmware product that is easily portable to any embedded software environment. Therefore, the WHD includes hooks for RTOS and TCP/IP network abstraction layers.
wcm	3.0.0	The Wi-Fi Connection Manager (WCM) is a library which helps application developers to manage Wi-Fi Connectivity. The library provides a set of APIs that can be used to establish and monitor Wi-Fi connections on Infineon platforms that support Wi-Fi connectivity.
whd-bsp-integration	2.1.0	The WHD library provides some convenience functions for connecting to a Board Support Package (BSP) that includes a WLAN chip. This library initializes the hardware and passes a reference to the communication interface on the board into WHD. It also sets up the LwIP based network buffers to be used for sending packets back and forth.
netxduo-network-interface-integration	1.0.0	This library is an integration layer that links the NetXDuo network stack with the underlying WHD. This library interacts with ThreadX, NetXDuo TCP/IP stack, and WHD. It contains the associated code to bind these components together.
lwip-network-interface-integration	1.0.0	This library is an integration layer that links the lwIP network stack with the underlying WHD and Ethernet driver. This library interacts with FreeRTOS, lwIP TCP/IP stack, WHD, and Ethernet driver. It contains the associated code to bind these components together.

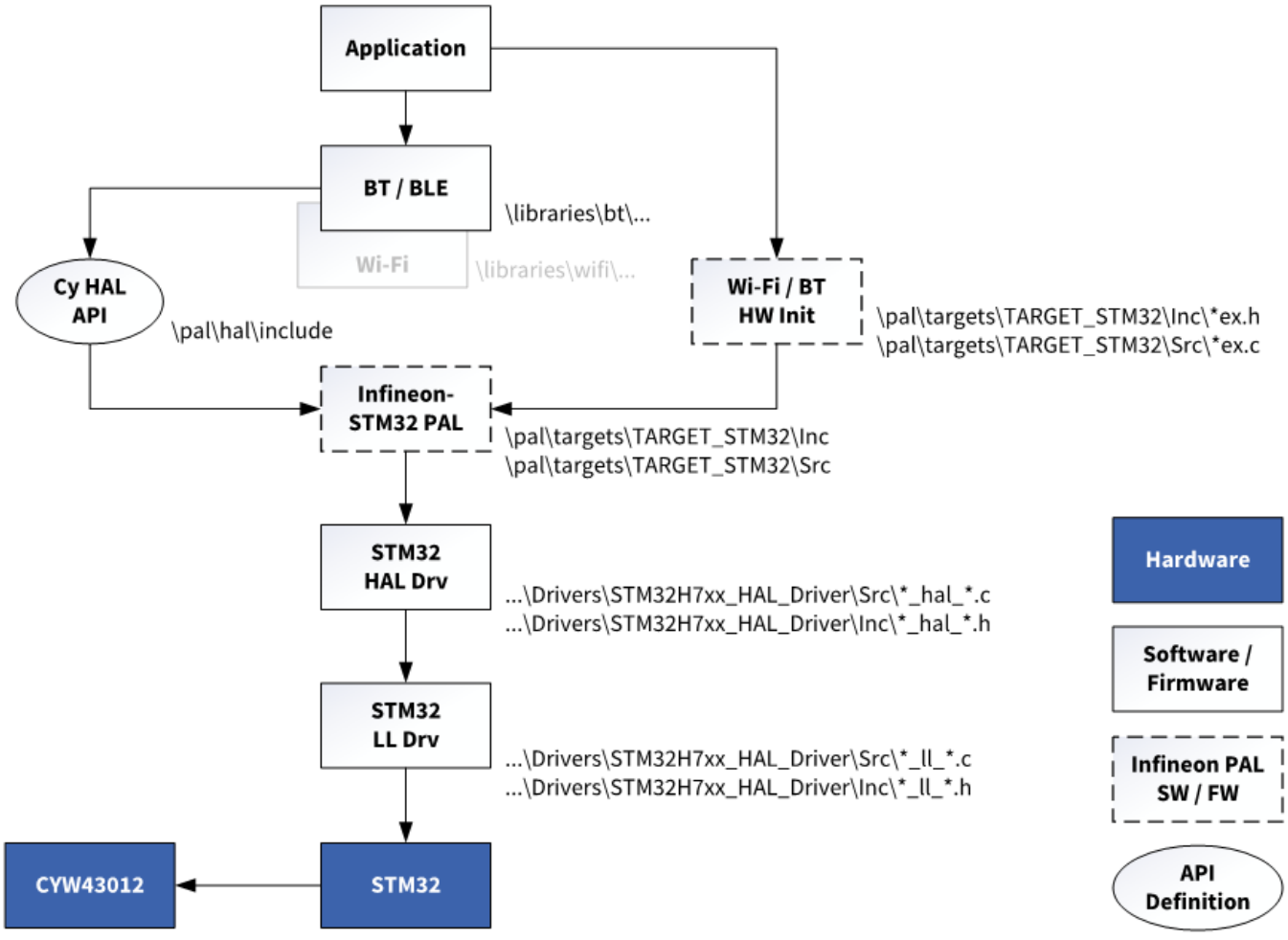
Component name	Version	Details
lwip-freertos-integration	1.0.0	This library contains the FreeRTOS dependancies needed by the Lightweight open-source TCP/IP stack, version: 2.1.2 to execute. See the https://savannah.nongnu.org/projects/lwip/ web site for details.
wifi-mfg-test	3.0.1	The WLAN Manufacturing Test Middleware application is used to validate the WLAN firmware and radio performance of the Wi-Fi device. The mfg-test middleware repo can accept the serial input byte stream from the Mfg Test application and transform the contained commands into IOVAR/IOCTL messages to the wlan firmware. It can get the response from the wlan firmware (if expected), and transport it back to the 'wl tool' running on the host.

1.1 Infineon-STM32 Platform Adaptation Layer (PAL)

The Infineon-STM32 PAL is based on the STM32 Driver MCU Component HAL, and it offers the minimum set of (required) APIs for Infineon-STM32 PAL. The supported HAL versions are:

STM32Cube HAL package	STM32Cube MCU verified package version
STM32H7 Series	1.10.0
STM32L5 Series	1.4.0

The PAL integrates the STM32 HAL APIs underneath the Infineon HAL APIs expected by the Infineon Connectivity Libraries. The figure below shows the architectural intent of the Infineon-STM32 PAL:



We created the Infineon-STM32-PAL to meet the following guidelines:

Developers will continue to use STM32CubeMX and/or STM32 HAL APIs to configure STM32 MCU hardware.

Developers will communicate to the PAL what STM32 hardware that they have selected and configured for communicating with a CYW43xxx via an initialization API.

Infineon-STM32 PAL adapts only the minimum set of Infineon HAL APIs to STM32 HAL in order to communicate and control Infineon's CYW43xxx Connectivity device(s).

The Infineon PAL layer behaves like the Infineon HAL as much as possible to minimize impact to the Infineon libraries. The Infineon PAL adapts the following STM32 HAL Drivers:

- GPIO
- LPTimer
- SDIO
- SPI
- TRNG
- UART

1.2 Supported STM32 MCUs

STM32H7xx

STM32L5xx

1.3 Supported STM32 boards

STM32H747I-DISCO Discovery kit

STM32L562E-DK

1.4 Supported connectivity modules

Infineon’s CYW43xxx Wi-Fi-Bluetooth combo chip family:

CYW43012

CYW43439 / CYW43438 / CYW4343W

CYW4373 / CYW4373/E

1.5 Compatible software

STM32 CubeMX 6.5.0

STM32 CubeIDE 1.9.0

IAR EWARM 8.50.4

2 Download/install/import expansion pack

2.1 Downloading the pack

Download the expansion pack from GitHub:

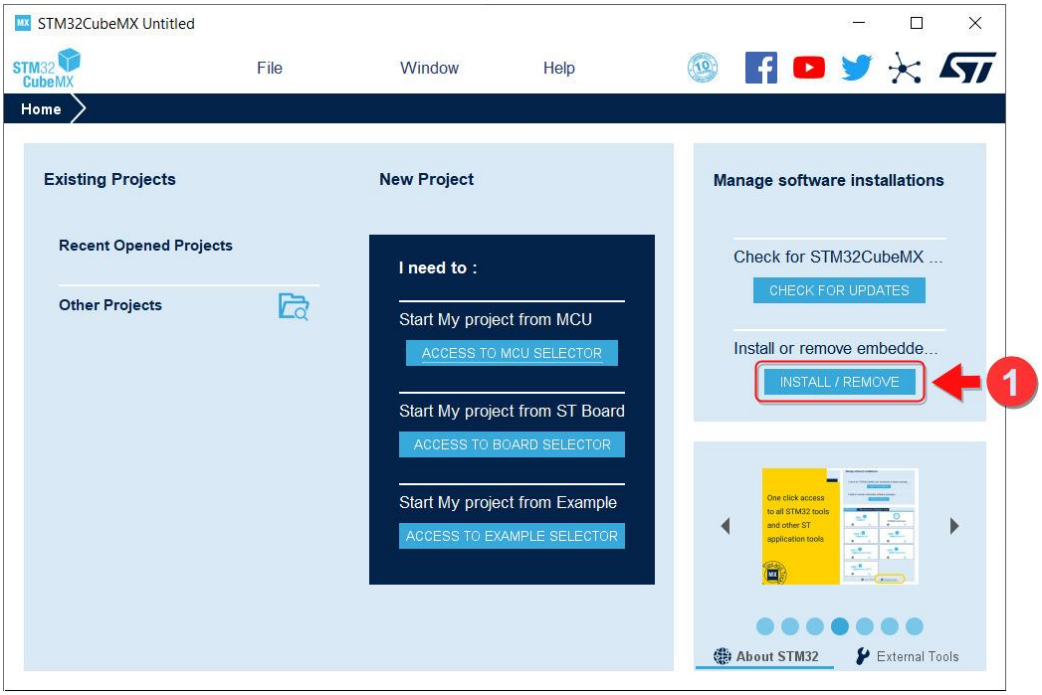
https://github.com/Infineon/stm32-connectivity/releases/tag/release-v1.2.0

2.2 Installing/importing the pack

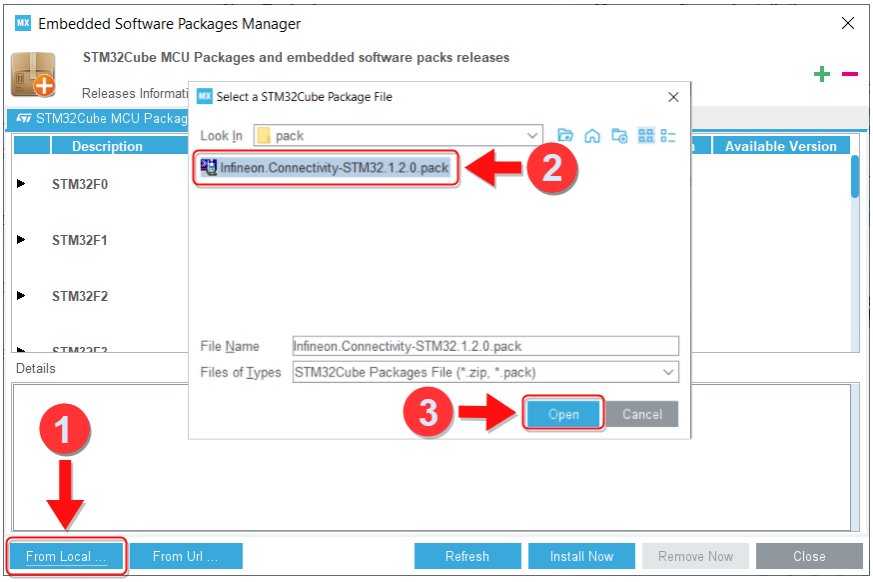
2.2.1 Add from local file

Perform these steps to add the expansion pack to the STM32 development environment:

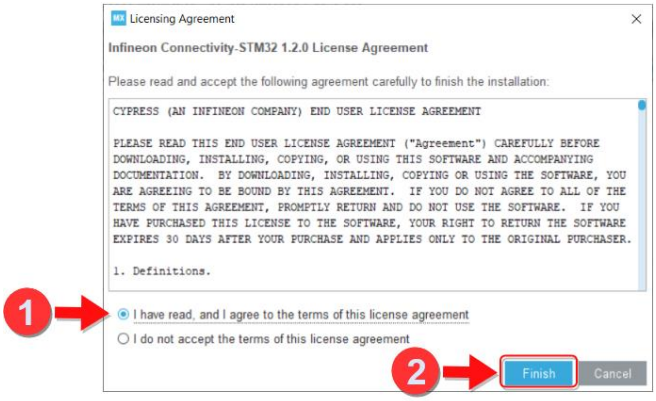
1. Run the STM32CubeMX tool.
2. Navigate to **Home > Manage software installations** and select **Install/Remove**.



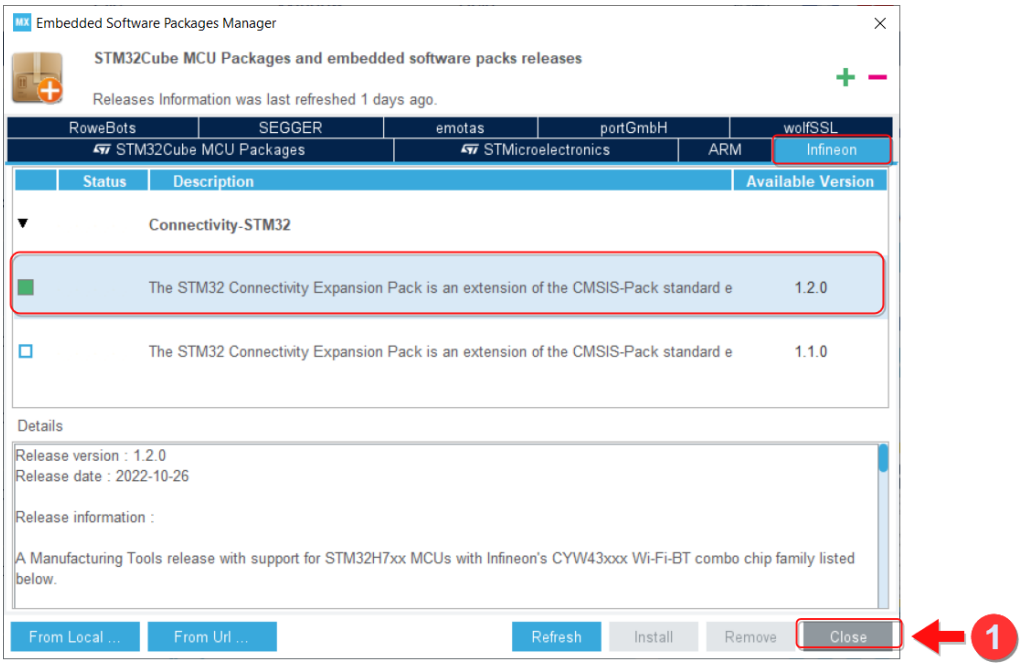
3. Select **From Local...**, navigate to the downloaded pack file, and select **Open**.



4. Accept the license agreement and select Finish.

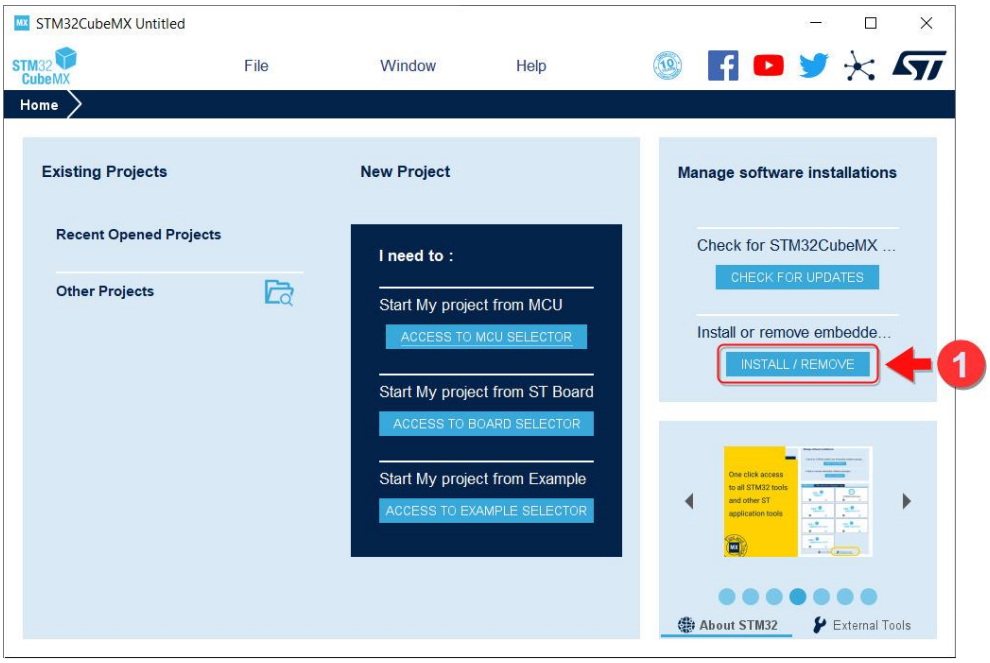


5. The tool shows an **Infineon** tab with the installed Expansion Pack displayed. Click **Close**.



2.2.2 Add the Pack from URL

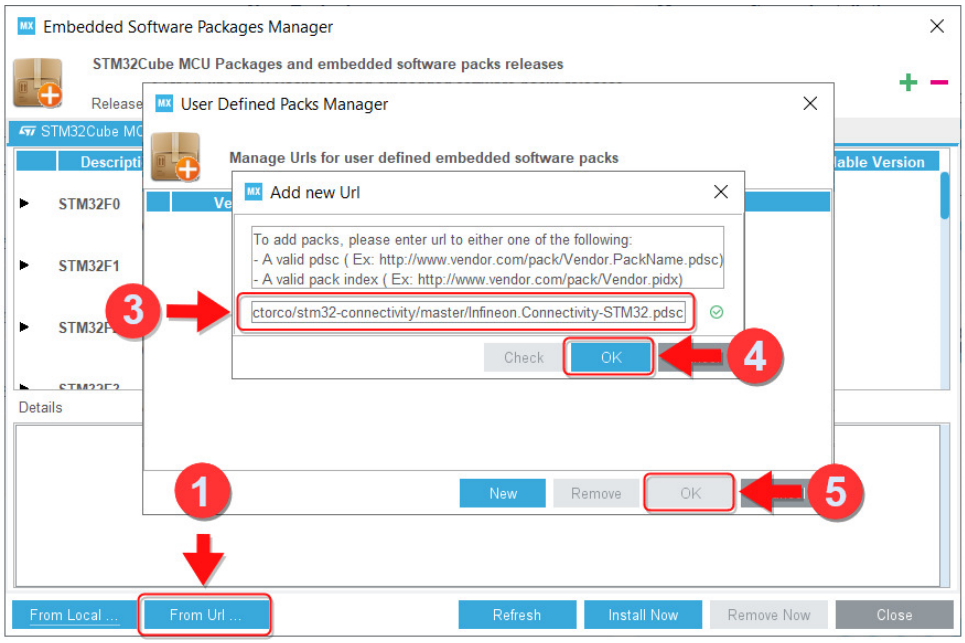
1. Run the STM32CubeMX tool.
2. Navigate to **Home > Manage software installations** and select **Install/Remove**.



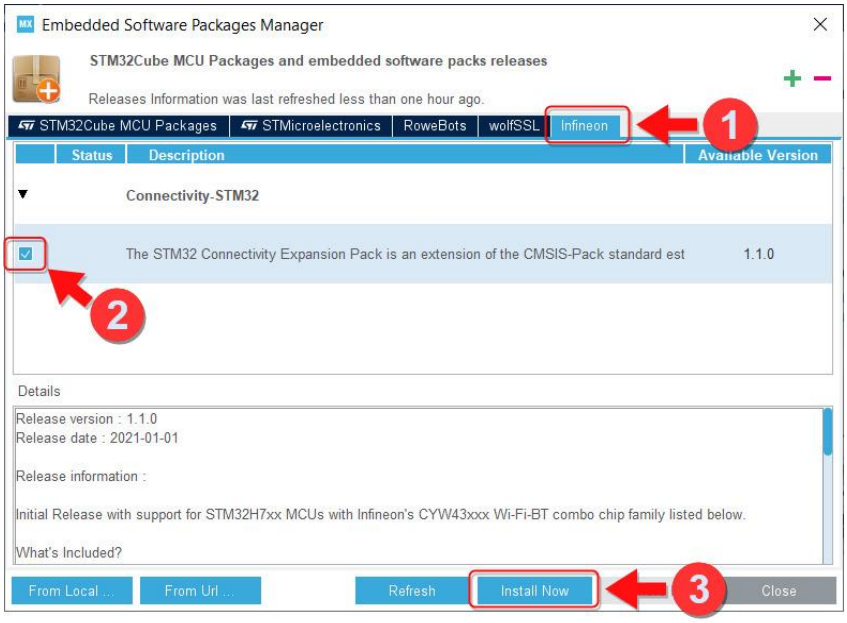
3. Select From URL…
4. Select **New (URL)**.
5. Input the Github URL to PDSC-file:

   https://raw.githubusercontent.com/cypresssemiconductorco/stm32-connectivity/master/Infineon.Connectivity-STM32.pdsc

6. Click **Check** and **OK** if check is successful.
7. Check just added URL and confirm with **OK** button.



8. In Software Package Manager check a pack and press **Install Now** to start online installation.



3 Hardware setup

3.1 Using STM32H747 DISCO kit

STM32H747 Disco Kit setup requires three discrete boards to create a setup where an STM32H747 hosts Infineon's CYW43xxx connectivity device. The three boards and links are:

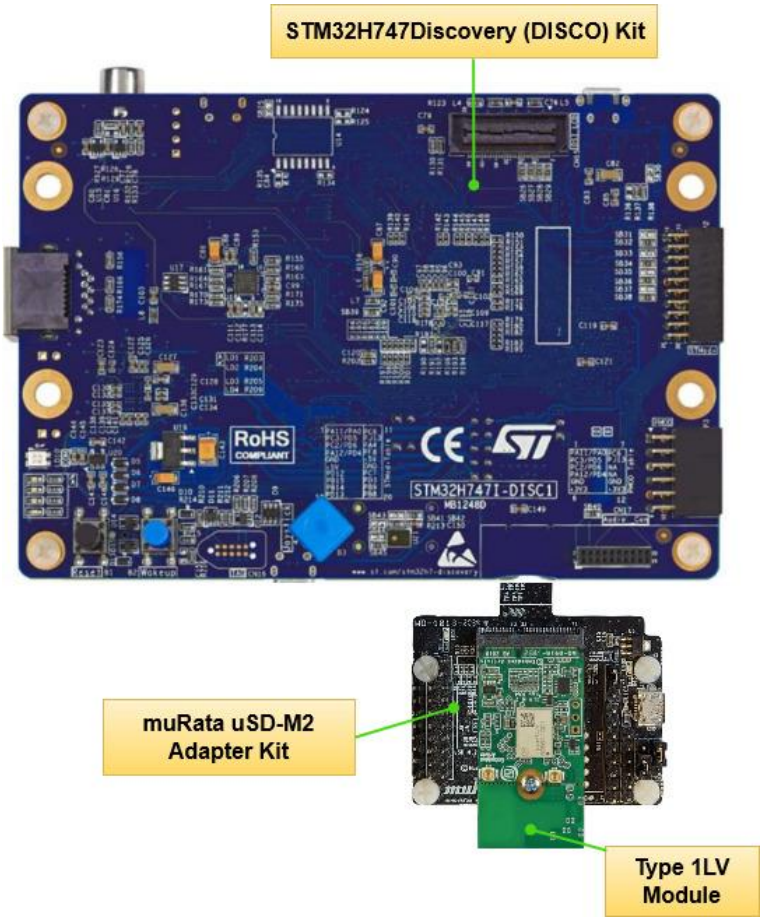
STM32H747 Discovery (DISCO) Kit: The STM32H747I-DISCO Discovery kit is a complete demonstration and development platform for STMicroelectronics STM32H747XIH6 microcontroller, designed to simplify user application development.

muRata uSD-M2 Adapter Kit (rev B1): muRata's uSD-M.2 Adapter Kit with Embedded Artists' Wi-Fi/Bluetooth M.2 Modules enable users with a simple plug-in solution. The Embedded Artists' Wi-Fi/Bluetooth M.2 Modules are based on Murata modules using Infineon's Wi-Fi/Bluetooth chipsets.

Current Wi-Fi/Bluetooth EVB support include

- Murata Type 1DX M.2 (CYW4343W)
- Type 1MW (CYW43455)
- Type 1LV M.2 (CYW43012)

Embedded Artists 1LV M.2 Module: Embedded Artists Type 1LV M.2 EVB is designed to work with the Murata uSD-M.2 Adapter.



3.1.1 Set up type 1LV M.2 module

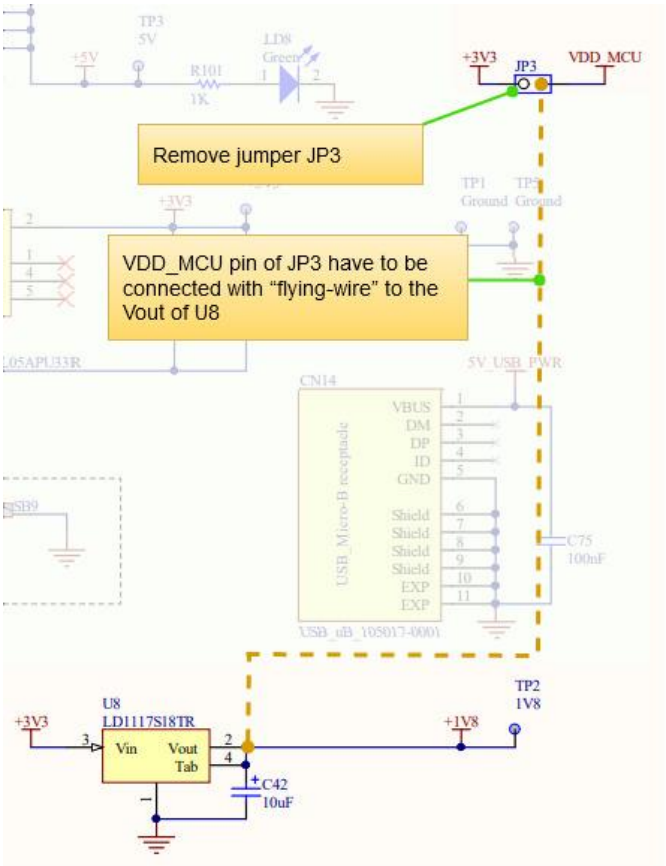
Model	Embedded Artists 1LV M.2 Module
Features	802.11 a/b/g/n/ac-friendly™ and Bluetooth/LE 5.0 SDIO 3.0 interface, SDR40@80MHz Chipset: Infineon CYW43012
Datasheet	1LV M.2

3.1.1.1 Board preparations

The 1LV module operates at 1.8 V VIO only (chipset limitation). The following preparation on STM32H747 DISCO Kit and muRata uSD-M2 Adapter are required:

1. Modify STM32H747 Disco Kit to operate on 1.8 V.

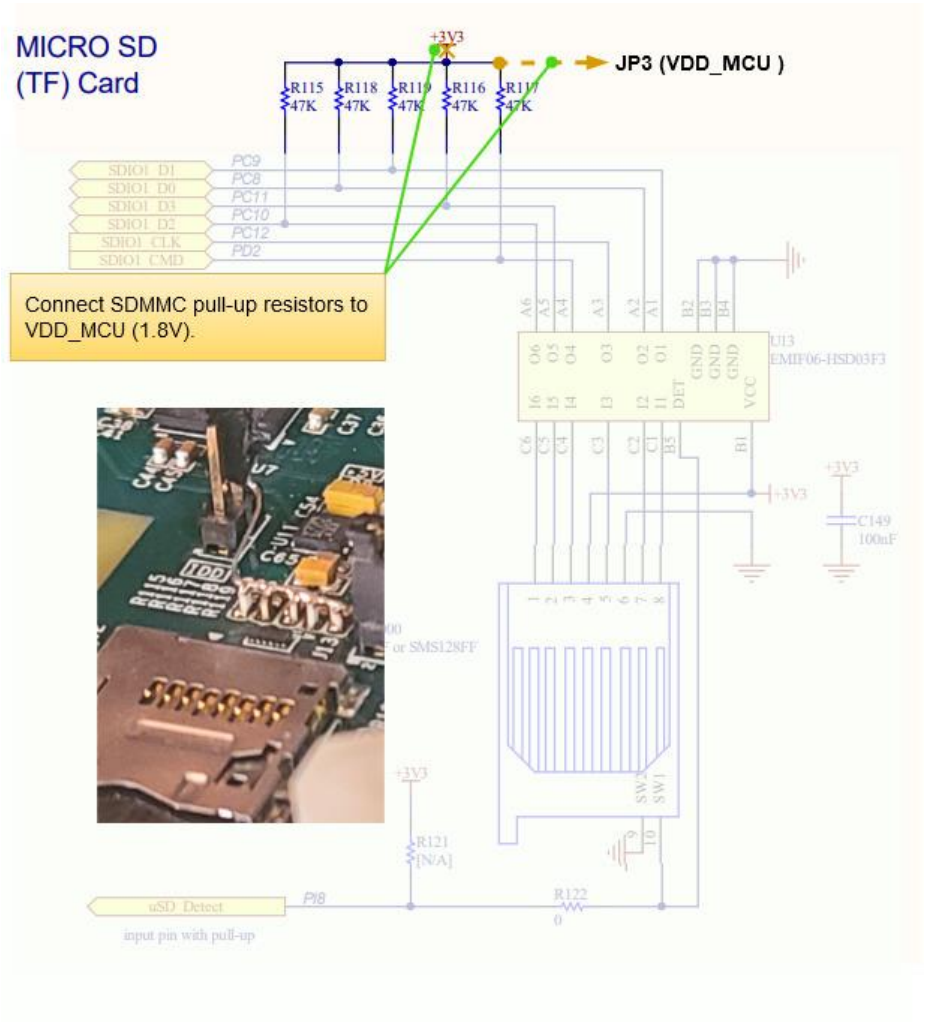
   Remove the jumper JP3 and connect the VDD_MCU pin of JP3 with “flying-wire” to the Vout of U8 linear voltage regulator (which is effectively a 1.8 V source).



Note: Switching STM32H747 Disco Kit to operate on 1.8 V affects the functionality of external flash (MT25QL512ABB8ESF).

2. Connect SDMMC pull-up resistors to VDD_MCU (1.8V) on STM32H747 DISCO Kit.

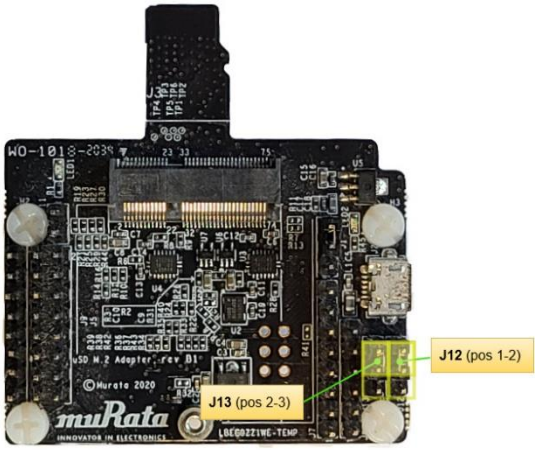
   SDMMC pull-up resistors R115-R119 must be unsoldered from the 3.3 V point and soldered vertically. The tops of these resistors have to be soldered to “flying-wire” and connected to JP3 at the side of VDD_MCU.



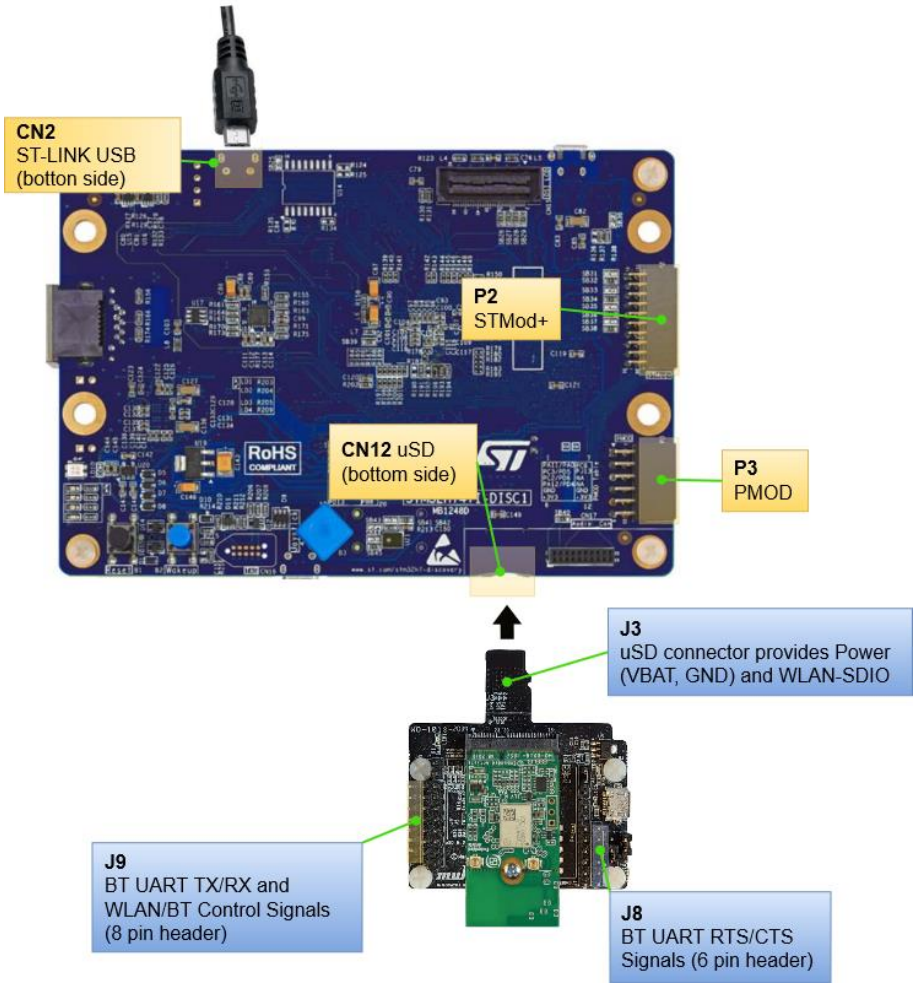
3. Modify muRata uSD-M2 Adapter to operate on 1.8V.

   To switch muRata uSD-M2 Adapter to 1.8V the following jumpers have to be configured:

   - J12 to pos 1-2 (M2 IO Voltage for 1.8V VDDIO)
   - J13 to pos 2-3 (Host IO Voltage for 1.8V)



3.1.1.2 Wire connections



Connection	Operation	STM32H747 Disco Kit		muRata uSD-M2 Adapter	Note
		Connector	STM32 GPIO		
VBAT (3.3V)	VCC	CN12		J3	VBAT, GND connected via microSD connector
GND	GND				
WL_REG_ON_HOST	Wi-Fi	P3.7 (PMOD#11)	PC6	J9.3	Enables/Disables WLAN core: Active High
WL_HOST_WAKE_HOST	Wi-Fi	P3.8 (PMOD#12)	PJ13	J9.5	WLAN Host Wake: Active Low (OOB IRQ)

Connection	Operation	STM32H747 Disco Kit		muRata uSD-M2 Adapter	Note
		Connector	STM32 GPIO		
SDIO	Wi-Fi	CN12	PC8, PC9, PC10, PC11, PC12, PD2	J3	uSD connector pin provides Power (VBAT, GND) and WLAN-SDIO (DATA1, DATA2, DATA3, Clock and Command)
UART RX	Bluetooth	P3.1 (PMOD#1)	PA11	J9.1	UART
UART TX	Bluetooth	P3.4 (PMOD#4)	PA12	J9.2	
UART CTS	Bluetooth	P2.8 (STmod+)	PB15	J8.3	
UART RTS	Bluetooth	P2.9 (STmod+)	PB14	J8.4	
BT_REG_ON	Bluetooth	P2.10 (STmod+)	PD13	J9.4	Enables/Disables Bluetooth core: Active High

3.1.2 Set up type 1DX M.2 module

Model	Embedded Artists 1DX M.2 Module
Features	802.11 b/g/n and Bluetooth/LE 4.2 SDIO 2.0 interface, SDR25@50MHz Chipset: Infineon CYW4343W
Datasheet	1DX M.2

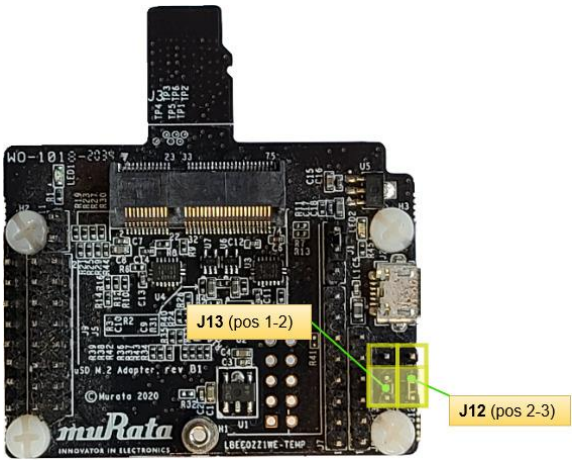
3.1.2.1 Board preparations

This module does not require the host to provide 1.8 V on the SDIO/UART GPIO. It can operate on 3.3V/1.8V. This makes board preparation simpler. Please see the following sections

1. Modify muRata uSD-M2 Adapter to operate on 3.3V.

   To switch muRata uSD-M2 Adapter to 3.3V the following jumpers have to be configured:

   - J12 to pos 2-3 (M2 IO Voltage for 3.3V VDDIO)
   - J13 to pos 1-2 (Host IO Voltage for 3.3V VDDIO)



3.1.2.2 Wire connections

The Type 1DXM module uses the same wire connections as Type 1LV modules. Refer to the Wire connections section for Type 1LV Modules.

3.2 Using STM32L562E-DK

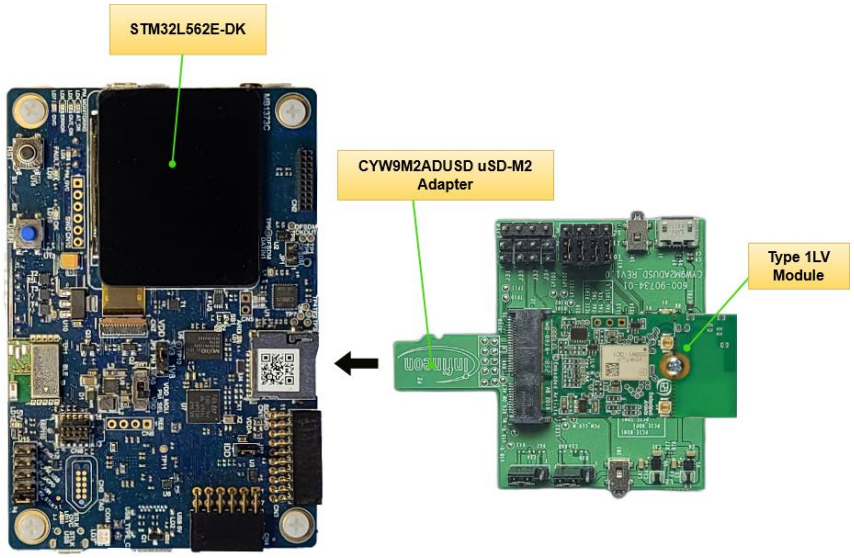
3.2.1 Set Up M.2 Module + CYW9M2ADUSD Adapter Kit for Wi-Fi and Bluetooth Connectivity

STM32H562E DK Kit setup for Bluetooth connectivity requires three discrete boards to create a setup where an STM32H562E hosts Infineon's CYW43xxx connectivity device. The three boards and links are:

STM32L562E-DK Discovery kit is a complete demonstration and development platform for Arm® Cortex®-M33 with Arm® TrustZone® and ARMv8-M mainline security extension core-based STM32L562QEI6QU microcontroller, with 512 Kbytes of Flash memory and 256 Kbytes of SRAM.

CYW9M2ADUSD Adapter Kit : adapter which allows you to connect M.2-based CYW43x connectivity modules into SD-card slot of a various DVKs and EVKs. Please contact sales for order questions.

Embedded Artists 1LV M.2 Module: Embedded Artists Type 1LV M.2 EVB is designed to work with the Murata uSD-M.2 Adapter.



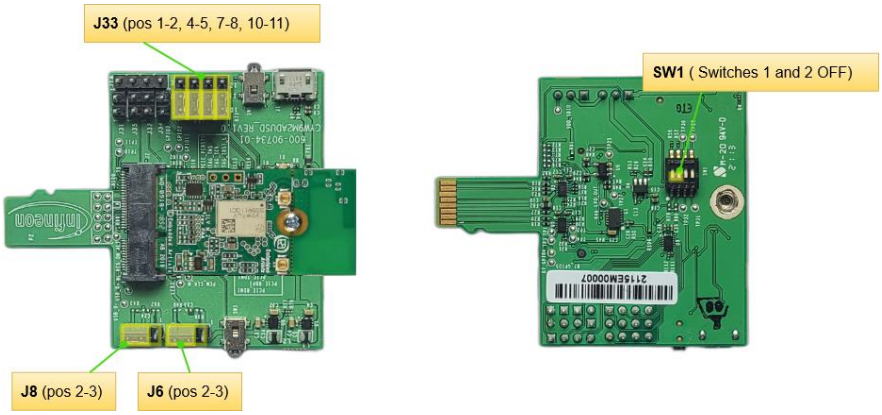
3.2.1.1 Board preparation

CYW9M2ADUSD Adapter requires to configure the following jumpers:

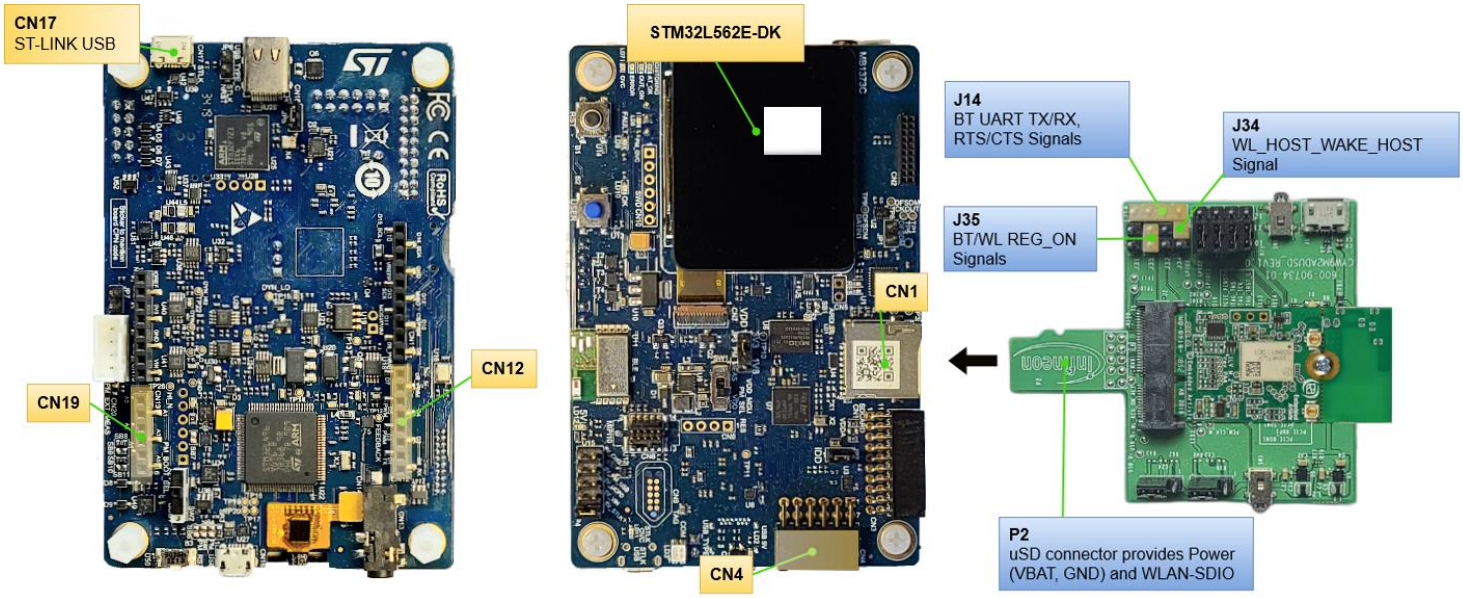
J6 and J8 to pos 2-3 (use 3.3V from VDD_SDIO)

J33 to use 1.8V level shifters for UART

SW1 – switches 1 and 2 in OFF position



3.2.1.2 Wire connections



Connection	Operation	STM32L562E-DK		CYW9M2ADUSD Adapter	Note
		Connector	STM32 GPIO		
VBAT (3.3V)	VCC	CN1		P2 (uSD Connection)	VBAT, GND connected via microSD connector
GND	GND				
WL_REG_ON_HOST	Wi-Fi	CN4.7	PF5	J35.1	Enables/Disables WLAN core: Active High
WL_HOST_WAKE_HOST	Wi-Fi	CN4.1	PB13	J34.1	WLAN Host Wake: Active Low (OOB IRQ)
SDIO	Wi-Fi	CN1	PC8, PC9, PC10, PC11, PC12, PD2	P2 (uSD Connection)	uSD connector pins: provides Power (VBAT, GND) and WLAN-SDIO (DATA0, DATA1, DATA2, DATA3, Clock and Command)
BT_REG_ON	Bluetooth	CN12.5	PF4	J35.2	Enables/Disables Bluetooth core: Active High
UART RX	Bluetooth	CN19.6	PC5	J14.2 (TX)	UART (USART3)
UART TX	Bluetooth	CN12.1	PB10	J14.1 (RX)	
UART CTS	Bluetooth	CN12.3	PD11	J14.4 (RTS)	
UART RTS	Bluetooth	CN12.4	PD12	J14.3 (CTS)	

4 Using example projects

We provide the following example projects to get started using the pack:

- Wi-Fi Scan
- Wi-Fi onboarding with Bluetooth® LE
- Azure RTOS NetXDuo Wi-Fi UDP echo server

4.1 Wi-Fi Scan

This example demonstrates how to configure different scan filters provided in the Wi-Fi Connection Manager (WCM) middleware and scan for the available Wi-Fi networks.

The example initializes the Wi-Fi device and starts a Wi-Fi scan without any filter and prints the results on the serial terminal. The example starts a scan every three seconds after the previous scan completes.

This example demonstrates how an STM32H7 can be used to host CYW43xxx connectivity devices.

4.1.1 Hardware

Refer to the section on the STM32 hardware configuration descriptions as appropriate:

Using STM32H747 DISCO Kit

4.1.2 Other software

Install a terminal emulator if you do not have one. Instructions in this document use Tera Term.

4.1.3 Project components

The following are the only components used in this project:

- Wifi/network-interface (configured as LWIP)
- Wifi/wifi-host-driver (WHD)
- Wifi/wcm
- Wifi/whd-bsp-integration
- Wifi/connectivity-utilities
- Wifi/LwIP
- Platform/pal (PAL, HAL, core-lib)
- Platform/abstraction-rtos (configured for the FreeRTOS kernel)
- Platform/device (configured as CYW43012)

4.1.4 Example project start/import

You can open the Wi-Fi Scan example by copying the example from the Pack to an appropriate location. Once you have copied the example, you can then open it in STM32CubeMX and export to your IDE using the following steps:

1. Copy the code example from the pack directory to your local directory.

   The default path for installed packs is:

C:\Users\<USER>\STM32Cube\Repository\Packs

Copy the wifi_scan example from the appropriate directory. For instance, for STM32H747I-DISCO:

C:\Users\<USER>\STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\1.2.0\Projects\STM32H747I-DISCO\Applications\wifi_scan

Paste into your working folder. For example:

C:\Users\<USER>\STM32Cube\Example

2. Open *wifi_scan.ioc* file in the root folder of project.

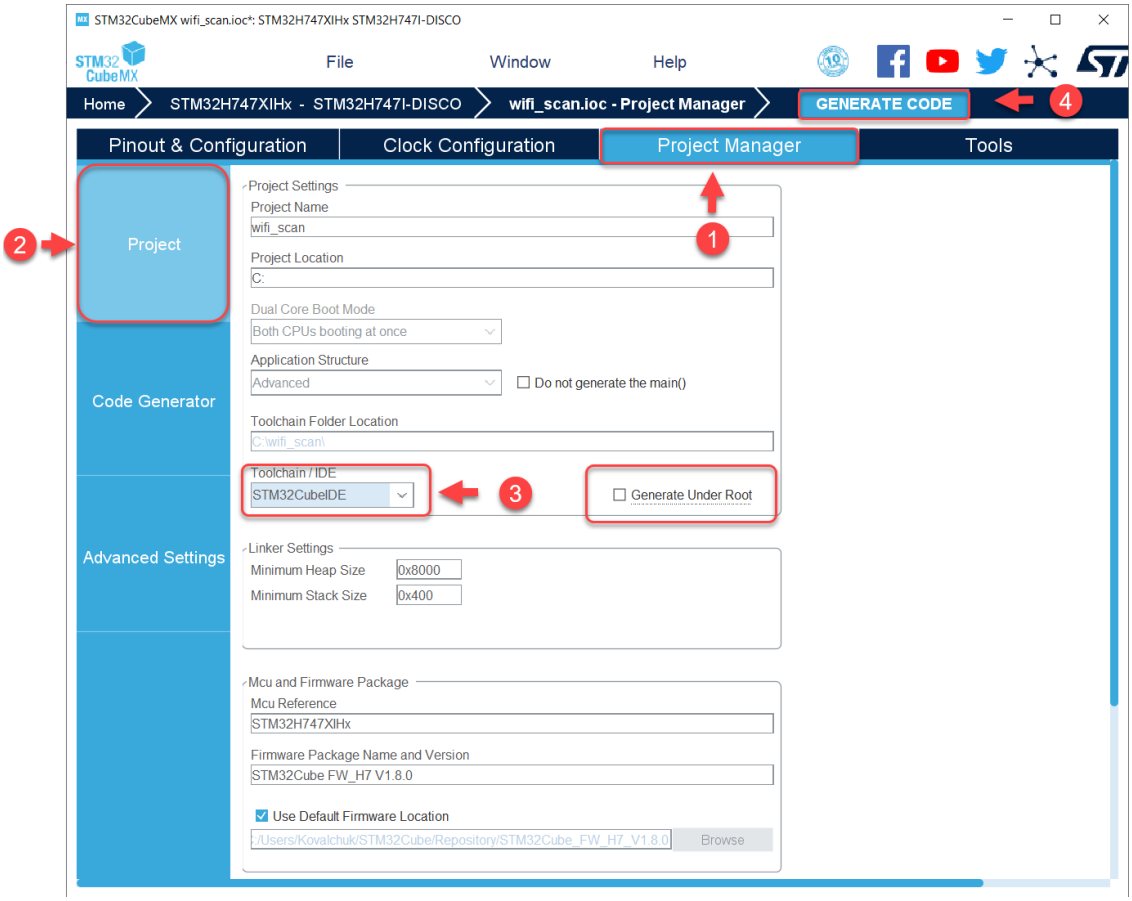
   C:\Users\<USER>\STM32Cube\Example\wifi_scan\wifi_scan.ioc

3. Click **OK** to accept.

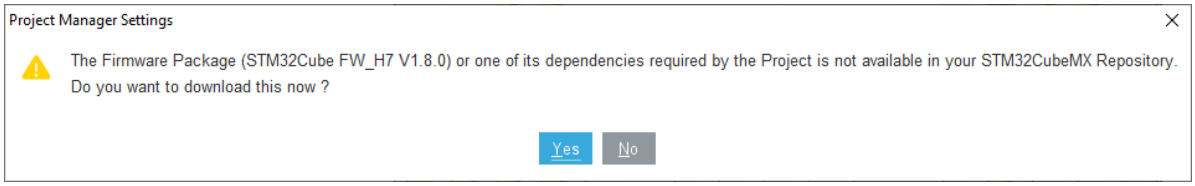
4.1.5 Generate code

Follow these steps to generate code:

1. Select the appropriate option under **Toolchain / IDE**.
2. Select the **Generate Under Root** check box.
3. Click **GENERATE CODE**.



If a message displays about missing packages, select **Yes**:

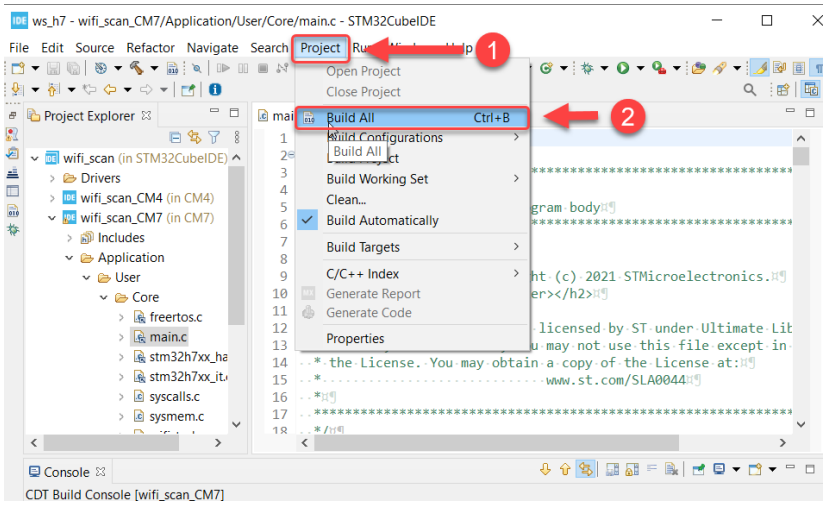


4. After the code is generated, you will see this dialog. Select **Open Project**.

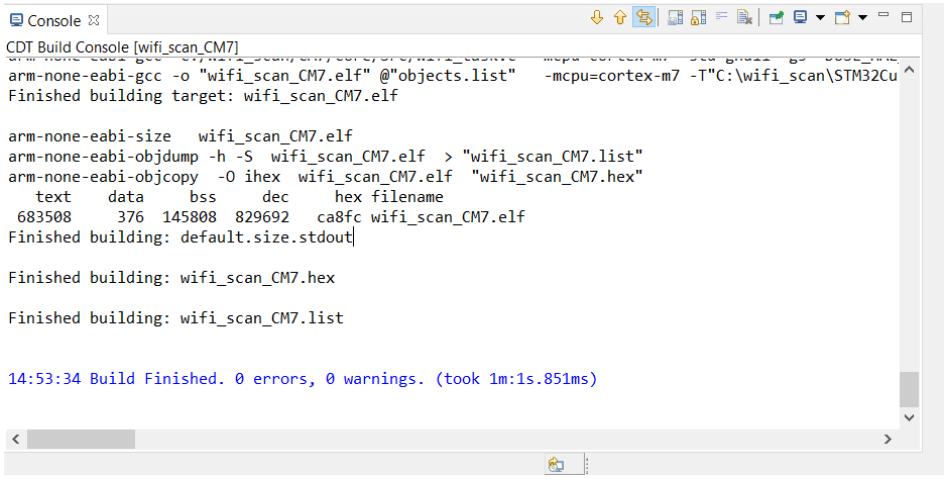
4.1.6 Build the project

The build step and expected output are illustrated here for each IDE.

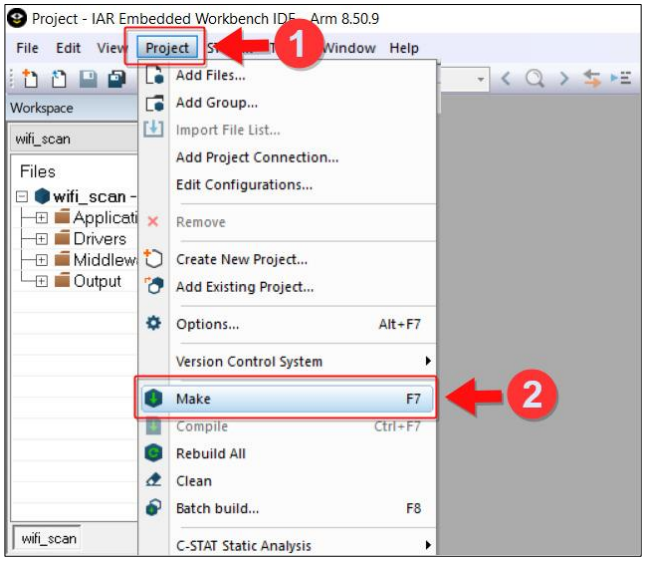
4.1.6.1 STM32CubeIDE:



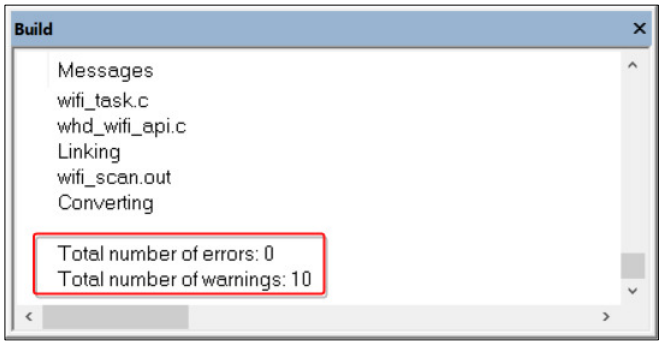
Example output from a successful build:



4.1.6.2 IAR EWARM:



The project should build without errors. There are 10 warnings in the lwIP library.



4.1.7 Project hardware setup

Refer to section Hardware Setup.

4.1.8 Terminal display

The terminal display is used by the application to provide status and network information.

You will need a terminal emulator such as Tera Term (https://ttssh2.osdn.jp/index.html.en) to display the output.

4.1.8.1 Serial terminal setup

The terminal interface is a virtual COM port which is part of the ST-LINK (CN2) USB connection. Terminal emulator configuration:

BaudRate: 115200

Data Length: 8 Bits

Stop Bit(s): 1

Parity: None

Flow control: None

4.1.8.2 Example output

```
****************** WiFi-Scan app ******************

    Insert CYW43xxx into microSD card slot

    Push blue button to continue...

    CYW43xxx detected

WLAN MAC Address : E8:E8:B7:9F:CC:EAWLAN Firmware : wl0: Sep  9 2020 01:22:10 version 13.10.271.253
(c4c4c7c CY) FWID 01-79301becWLAN CLM            : API: 18.2 Data: 9.10.0 Compiler: 1.36.1 ClmImport:
1.34.1 Creation: 2020-09-09 01:19:03 WHD VERSION  : v1.93.0 : v1.93.0 : IAR 8050009 : 2020-12-21
13:24:03 +0530
-----------------------------------------------------------------------------------------------
  #                 SSID                 RSSI   Channel       MAC Address              Security
-----------------------------------------------------------------------------------------------
  1   Private                             -72      11      1C:AF:F7:26:8D:A8         WPA2_MIXED_PSK
  2   Private                             -73      11      74:DA:88:29:F2:27         WPA2_MIXED_PSK

-----------------------------------------------------------------------------------------------
  #                 SSID                 RSSI   Channel       MAC Address              Security
-----------------------------------------------------------------------------------------------
  1   Private                             -68      11      74:DA:88:29:F2:27         WPA2_MIXED_PSK
  2   Private                             -73      11      1C:AF:F7:26:8D:A8         WPA2_MIXED_PSK
```

4.2 Wi-Fi onboarding with Bluetooth LE

This example uses the STM32H7 MCU to communicate with the CYW43xxx combo devices and control the Wi-Fi and BLE functionality. It uses BLE on the combo device to help connect the Wi-Fi to the AP.

In this example, BLE provides a mechanism for the device to connect to a Wi-Fi AP by providing the Wi-Fi SSID and password in a secure manner. The Wi-Fi credentials are stored in EEPROM so that the device can use this data upon reset to connect to an AP without requiring BLE intervention. Note that the data stored in the EEPROM is unencrypted.

The Wi-Fi SSID and password are exchanged using custom GATT service and characteristics. There is a third custom characteristic, which gives the command to connect and disconnect. The Wi-Fi password is write-only; the device needs to be paired before this characteristic can be written.

BLE GATT Custom Service
This example uses custom GATT service and characteristics to communicate with the BLE GATT client. The files cycfg_gatt_db.c and cycfg_gatt_db.h contain the GATT DB.

The following custom characteristics are used in this example:

WiFi SSID: Provides the Wi-Fi SSID from BLE GATT client to the server. The maximum size is 32 as Wi-Fi limits the SSID name to 32 characters.

WiFi Password: Provides the Wi-Fi password from the BLE GATT client to the server. The minimum size is 8 because Wi-Fi encryption requires a minimum of 8 characters for password.

WiFi Connect: A boolean characteristic that is used to connect and disconnect from the Wi-Fi AP. This has a Cleint Characteristic Configuration Descriptor (CCCD) attached with it. Whenever there is a successful connection it will send a notification value of 1 otherwise it will send a notification value of 0 if notifications are enabled.

4.2.1 Hardware

Refer to section the STM32 hardware configuration descriptions as appropriate: Using STM32H747 DISCO Kit

4.2.2 Other software

This code example requires two devices: Host (Mobile Phone or PC) and a Target (STM32H747 DISCO Kit).

1. For the Host, download and install the CySmart™ app for iOS or Android. Scan the following QR codes from your mobile phone to download the CySmart app:

iOS

Android

2. Install a terminal emulator if you don't have one. Instructions in this document use Tera Term.

4.2.3 Project components

The following are the components used in this project:

- Wifi/network-interface (configured as LWIP)
- Wifi/wifi-host-driver (WHD)
- Wifi/wcm
- Wifi/whd-bsp-integration
- Wifi/connectivity-utilities
- Wifi/LwIP
- Bluetooth/btstack
- Bluetooth/bluetooth-freertos
- Platform/pal (PAL, HAL, core-lib)
- Platform/abstraction-rtos (configured for the FreeRTOS kernel)
- Platform/device (configured as CYW43012)

4.2.4 Example project start/import

You can open the Wi-Fi Onboarding with BLE example by copying the example from the Pack to an appropriate location:

C:\Users\<USER>\STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\1.2.0\Projects\STM32H747I-DISCO\Applications\ble_wifi_onboarding

Once you have copied the example, you can then open it in STM32CubeMX and export to your IDE using the following steps from Example project start/import, Generate code, Build the project from Wi-Fi Scan example.

4.2.5 Project hardware setup

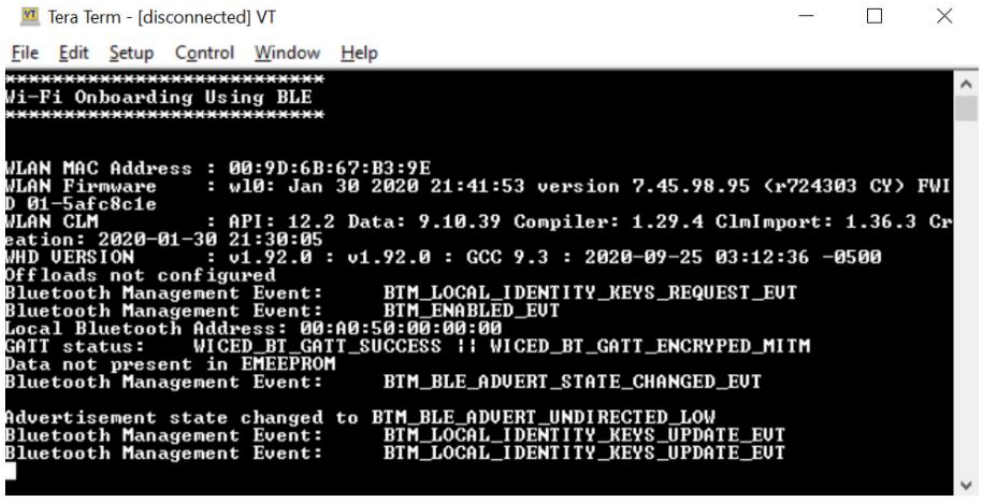
Refer to section Hardware Setup.

4.2.6 Operation

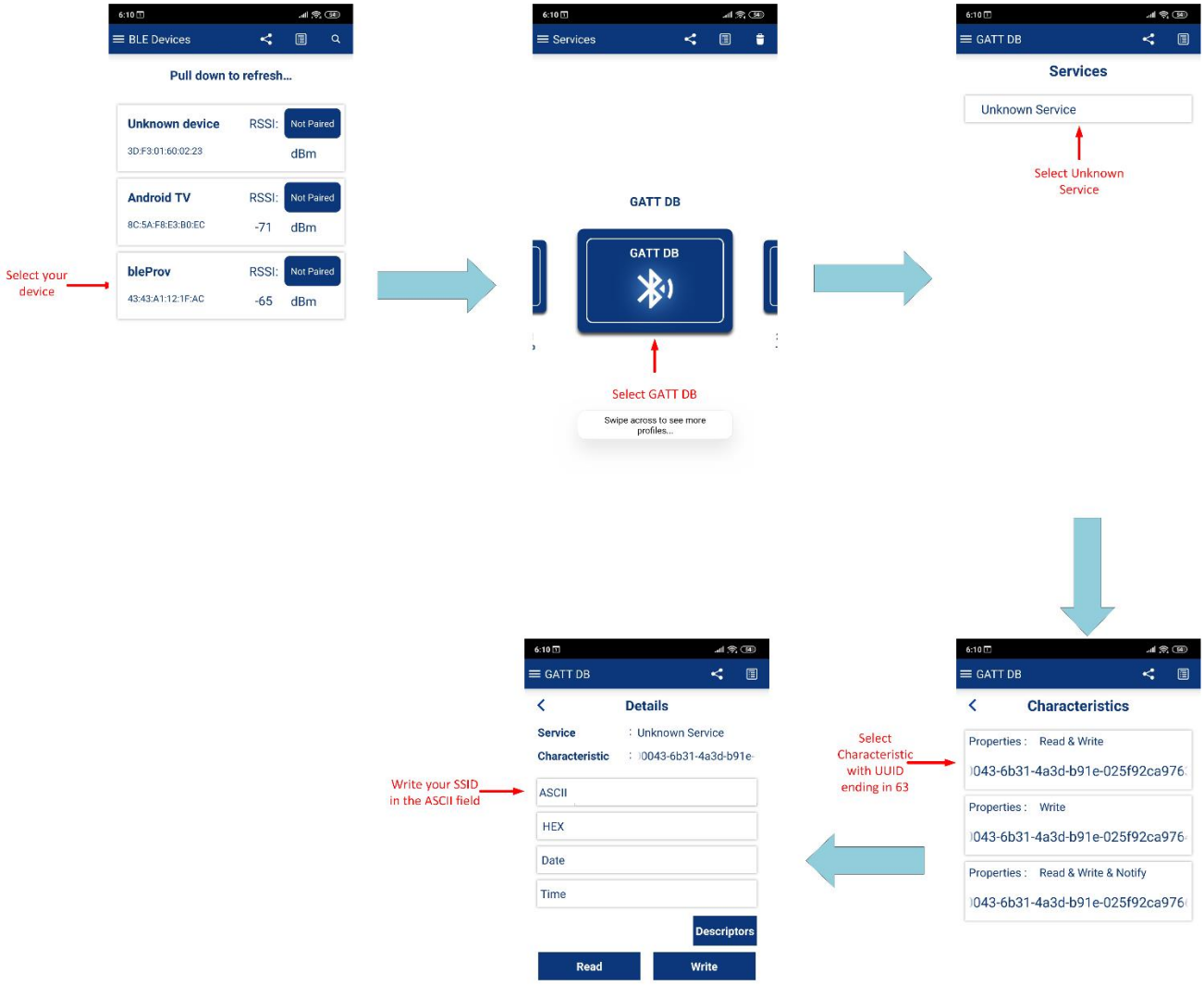
1. Connect the STM32H747 DISCO Kit to your PC.
2. Use your favorite serial terminal application and connect to the ST-LINK (CN2) COM port. Configure the terminal application to access the serial port using the following settings.

   Baud rate: 115200 bps; Data: 8 bits; Parity: None; Stop: 1 bit; Flow control: None; New line for receive data: Line Feed (LF) or Auto setting
3. Program the board.

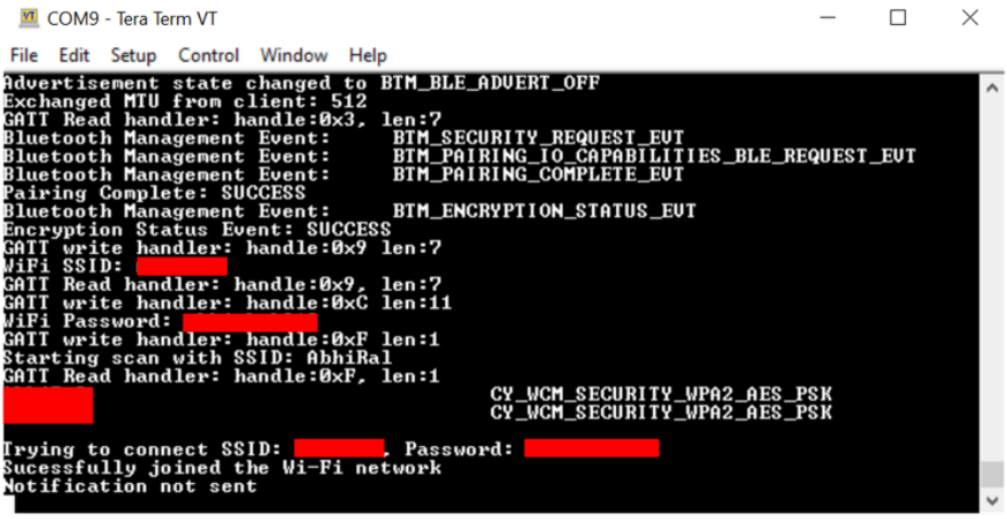
   After programming, the application starts automatically. Observe the messages on the UART terminal, and wait for the device to make all the required connections.



4. To test using the CySmart mobile app, do the following
   a. Turn ON Bluetooth on your Android or iOS device.
   b. Launch the CySmart app.
   c. Press the reset switch on the kit to start sending advertisements.
   d. Swipe down on the CySmart app home screen to start scanning for BLE Peripherals. Your device (“bleProv”) appears in the CySmart app home screen. Select your device to establish a BLE connection.
   e. Select the **GATT DB** Profile from the carousel view then select **Unknown Service**.
   f. Select the attribute with the UUID ending in 63. In the ASCII field, type your Wi-Fi SSID in string format. Do the same for password UUID ending in 64) as described above.



5. Select the attribute with the UUID ending in 65. Select **Notify**. Write hex value 1 to this characteristic to connect to the Wi-Fi network. If the connection is successful then the server will send a notification with the value 1 or with the value 0.



Once the Wi-Fi SSID and password are provided by the client it is stored in the EEPROM. To delete this data the user needs to press the User Button.

4.3 Azure RTOS NetXDuo Wi-Fi UDP echo server

This application provides an example of Azure RTOS NetX/NetXDuo stack usage. It shows you how to develop a NetX UDP server to communicate with a remote client using the NetX UDP socket API.

This example demonstrates how an STM32H7 can be used to host CYW43xxx connectivity devices.

4.3.1 Hardware

Refer to the section on the STM32 hardware configuration descriptions as appropriate: Using STM32H747 DISCO Kit

4.3.2 Other software

Install a terminal emulator if you don't have one. Instructions in this document use Tera Term.

Download echotool utility.

4.3.3 Project components

The following are the only components used in this project:

- Wifi/network-interface (configured as NetXDuo)
- Wifi/wifi-host-driver (WHD)
- Wifi/wcm
- Wifi/whd-bsp-integration
- Wifi/connectivity-utilities
- Bluetooth/btstack
- Bluetooth/bluetooth-freertos
- Platform/pal (PAL, HAL, core-lib)
- Platform/abstraction-rtos (configured for the ThreadX kernel)
- Platform/device (configured as CYW43012)

4.3.4 Example project start/import

You can open this example by copying the example from the Pack to an appropriate location:

C:\Users\<USER>\STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\1.2.0\Projects\STM32H747I-DISCO\Applications\wifi_netxduo

Once you have copied the example, you can then open it in STM32CubeMX and export to your IDE using the following steps from Example project start/import, Generate code, Build the project from Wi-Fi Scan example.

4.3.5 Project hardware setup

Refer to section Hardware Setup.

4.3.6 Operation

1. Connect the board to your PC using the provided USB cable through the ST-Link USB connector.
2. Modify the `WIFI_SSID` and `WIFI_PASSWORD` macros in *Application/User/NetXDuo/console_task.c* to match with those of the Wi-Fi network that you want to connect to.

3. Update the `DEFAULT_PORT` macro in *Application/User/NetXDuo/console_task.c*.
4. Open a terminal program and select the **ST-Link COM** port. Set the serial port parameters to 8N1 and 115200 baud.
5. Program the board using STM32CubeIDE or EWARM.

   After programming, the application starts automatically. Observe the messages on the UART terminal, and wait for the device to make the required connections.
6. Run the echotool utility on a windows console as following:

   ```
   # echotool.exe  <board IP address> /p udp  /r  <DEFAULT_PORT> /n 10 /d "Hello World"
   ```

   Example usage:

   ```
   echotool.exe 192.168.1.2 /p udp /r 6000 /n 10 /d "Hello World"
   ```

5 Manufacture tools

The following manufacture tool projects are included in the pack:

- Tester - Wi-Fi Bluetooth® Console
- WLAN manufacturing test application (Wifi-Mfg-Tester) for FreeRTOS
- Bluetooth Manufacturing Test Application for FreeRTOS

5.1 Tester - Wi-Fi Bluetooth® Console

This application integrates the command console library including Wi-Fi iPerf and Bluetooth® Low Energy functionality. You can use this application to characterize the Wi-Fi/Bluetooth® LE functionality and performance.

This example demonstrates how an STM32H7 can be used to host CYW43xxx connectivity devices.

5.1.1 Hardware

Refer to the section on the STM32 hardware configuration descriptions as appropriate:

Using STM32H747 DISCO Kit

5.1.2 Other software

Install a terminal emulator if you don't have one. Instructions in this document use Tera Term.

Setting up iPerf on the host:

- iPerf 2.0.13 (supported on Ubuntu, macOS, and Windows)
- Go to the iPerf installation directory and launch the terminal (command prompt for Windows, terminal shell for macOS or Ubuntu).

5.1.3 Project components

The following are the only components used in this project:

- Wifi/network-interface (configured as LWIP)
- Wifi/wifi-host-driver (WHD)
- Wifi/wcm
- Wifi/whd-bsp-integration
- Wifi/connectivity-utilities
- Wifi/secure-sockets
- Wifi/LwIP
- Bluetooth/btstack
- Bluetooth/bluetooth-freertos
- Platform/pal (PAL, HAL, core-lib)
- Platform/abstraction-rtos (configured for the FreeRTOS kernel)
- Platform/device (configured as CYW43012)
- MfgTools/command-console

5.1.4 Example project start/import

You can open this example by copying the example from the Pack to an appropriate location:

C:\Users\<USER>\STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\1.2.0\Projects\STM32H747I-DISCO\Applications\wifi_bt_tester

Once you have copied the example, you can then open it in STM32CubeMX and export to your IDE using the following steps from Example project start/import, Generate code, Build the project from Wi-Fi Scan example.

5.1.5 Project hardware setup

Refer to section Hardware Setup.

5.1.6 Operation

1. Connect the board to your PC usingthe provided USB cable through the ST-Link USB connector.
2. Modify the **WIFI_SSID** and **WIFI_KEY** macros in *Application/User/Core/console_task.c* to match with those of the Wi-Fi network that you want to connect to.
3. To join a Wi-Fi network of a specific band, update the **WIFI_BAND** macro in *Application/User/Core/console_task.c* as follows:

   CY_WCM_WIFI_BAND_2_4GHZ: 2.4-GHz band **CY_WCM_WIFI_BAND_5GHZ**: 5-GHz band
4. Configure the TCP window size in iPerf before building the application. See the command console library's Readme.md for instructions on how to configure the TCP window size.
5. Open a terminal program and select the ST-Link COM port. Set the serial port parameters to 8N1 and 115200 baud.
6. Program the board using STM32CubeIDE or EWARM. After programming, the application starts automatically. Observe the messages on the UART terminal, and wait for the device to make the required connections.
7. The application connects to the configured Wi-Fi access point (AP) and obtains the IP address. When the device is ready, the > prompt appears.
8. Run iPerf commands (client and server) against a remote peer device.
   - See Running iPerf client and server against a remote peer device.
9. Run Bluetooth® LE commands against a remote peer device.

5.1.7 Serial terminal setup

The terminal interface is a virtual COM port which is part of the ST-LINK (CN2) USB connection. Terminal emulator configuration:

BaudRate: 115200

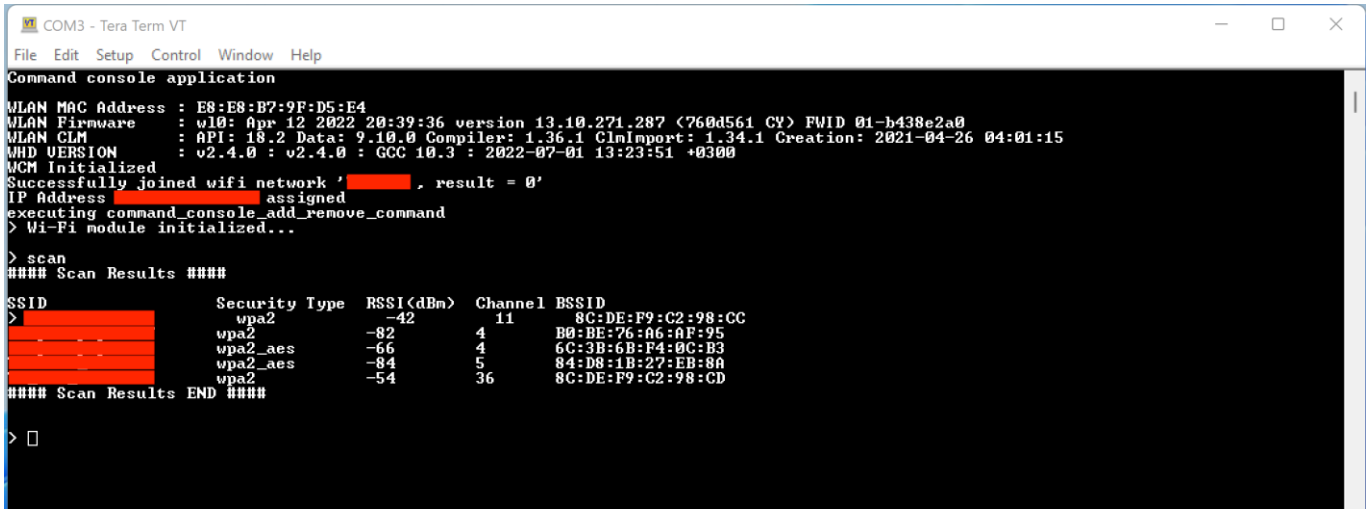
Data Length: 8 Bits

Stop Bit(s): 1

Parity: None

Flow control: None

5.1.8 Example output

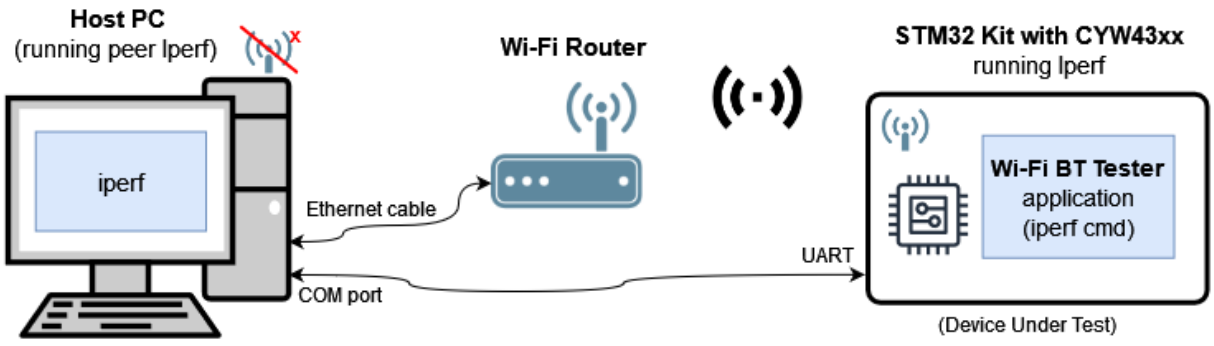


5.1.9 Iperf measurment

iPerf commands are used for measuring the Wi-Fi performance/throuьфрьghput. The iperf sends TCP/UDP data between two peer devices to compute the Wi-Fi performance/throughput.

5.1.9.1 iPerf setup

The following diagram shows the exact setup that should be used for measuring the Wi-Fi performance/throughput of a STM32 device using iperf.



5.1.9.2 iPerf commands for Wi-Fi throughput measurement

Enter the following commands on the STM32 device (DUT) after the device boots up and connects to the Wi-Fi network. This section provides only the commands to be run on the DUT. When the 'client *iperf* command' runs on the DUT, the 'server *iperf* command' should run on the host PC (as shown in the iPerf Setup diagram), and vice versa.

1. Start iPerf as a TCP server:

   ```
   iperf -s
   ```

   Note: On the peer iPerf device (host PC), start iPerf as a TCP client to send the TCP data.

2. Start iPerf as a TCP client:

   ```
   iperf -c <server_ip_addr> -t <time in sec>
   ```

   Note: On the peer iPerf device (host PC), start iPerf as a TCP server.

   Sample command:

   ```
   iperf -c 192.168.0.100 -t 60
   ```

3. Start iPerf as a UDP server:

   ```
   iperf -s -u
   ```

   Note: On the peer iPerf device (host PC), start iPerf as a UDP client to send the UDP data.

4. Start iPerf as a UDP client:

   ```
   iperf -c <server_ip_addr> -t <time in sec> -u -b <band width>
   ```

   Note: On the peer iPerf device (host PC), start iPerf as a UDP server.

   Sample command:

   ```
   iperf -c 192.168.0.100 -t 60 -u -b 50M
   ```

5.1.9.3 Results

STM32H747 DISCO + CYW43012

TCP/ UDP	Throughput, Mbit/s		Command
	2.4G	5G	
TCP TX	35.3	42.7	iperf -c <ip> -t 60
TCP RX	35.7	39.2	iperf -s
UDP TX	52.4	52.4	iperf -c <ip> -t 60 -u -b 50M
UDP RX	50.0	50.0	iperf -s -u

Test configuration: Iperf app run on STM32H747 CM7/400Mhz, GCC, Wi-Fi router: Asus RT-AX56U.

STM32L5-DK + CYW43012

TCP/ UPD	Throughput, Mbit/s		Command
	2.4G	5G	
TCP TX	20.2	20.5	iperf -c <ip> -t 60
TCP RX	20.6	20.8	iperf -s
UDP TX	31.0	31.1	iperf -c <ip> -t 60 -u -b 50M
UDP RX	25.7	24.7	iperf -s -u

Test configuration: Iperf app run on STM32L5 CM33/110Mhz, GCC, router: Wi-Fi Asus RT-AX56U.

5.2 WLAN manufacturing test application (Wifi-Mfg-Tester) for FreeRTOS

The Wifi-Mfg-Tester is used to validate the WLAN firmware and radio performance of Wi-Fi chips.

The Wifi-Mfg-Tester acts as a transport layer between the host "wl tool" and the WLAN firmware, and receives the commands from the wl tool and forwards them to the WLAN firmware using IOVAR/IOCTL commands. It also relays the response received back from the WLAN firmware.

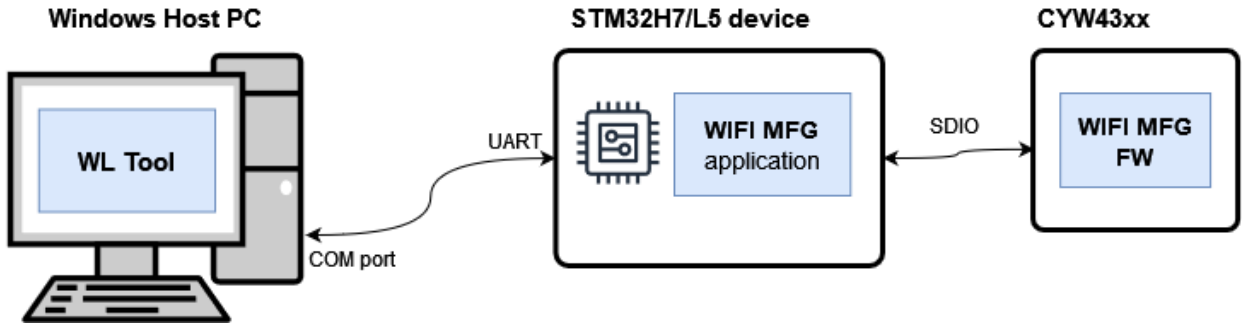
The wl tool binaries for testing the WLAN firmware are also included in this application repository.

This example demonstrates how an STM32H7 can be used to host CYW43xxx connectivity devices.

5.2.1 Hardware

Refer to the section on the STM32 hardware configuration descriptions as appropriate: Using STM32H747 DISCO Kit

Test setup are shown below:



5.2.2 Other software

Install a terminal emulator if you don't have one. Instructions in this document use Tera Term.

This application requires the WL tool running on a Windows PC

- The pre-built executables for the WL tool are available in the *wl-tool-bin*.

5.2.3 Project components

The following list shows the only components used in this project:

- Wifi/wcm
- Wifi/wifi-mw-core
- Wifi/wifi-host-driver (WHD)
- Wifi/whd-bsp-integration
- Wifi/connectivity-utilities
- Wifi/LwIP
- Platform/pal (PAL, HAL, core-lib)
- Platform/abstraction-rtos (configured for the FreeRTOS kernel)
- Platform/device
- MfgTools/wifi-mfg-test

5.2.4 Example project start/import

You can open the example by copying this example from the Pack to an appropriate location:

C:\Users\<USER>\STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\1.1.0\Projects\STM32H747I-DISCO\Applications\wifi_mfg_tester

Once you have copied the example, you can then open it in STM32CubeMX and export to your IDE using the following steps from Example project start/import, Generate code, Build the project from Wi-Fi Scan example.

5.2.5 Project hardware setup

Refer to section Hardware Setup.

5.2.6 Operation

1. Go to the WL tool directory:

```
# cd wl-tool-bin
```

2. Reset the board by pressing the Reset button.
3. Run the command on Windows host for the WLAN chip on the target board:

   ```
   wl43012C0.exe --serial <port> ver
   ```

   For example:

   ```
   #wl43012C0.exe --serial 5 ver
   cmd resp: 7/19/2017 build 0
   wl0: Jan 11 2022 21:32:24 version 13.10.271.280 (c32ff79 CY WLTEST) FWID 01-
   3566e923
   ```

4. Observe the output of the command.

   The list of WL commands which can be retrieved by typing `--help`. Partial output of the command and display is as follows:

   ```
   # wl43012C0.exe --serial 5 –help

   Usage: wl43012C0.exe [-a|i <adapter>] [-h] [-d|u|x] <command> [arguments]
       -h          this message and command descriptions
       -h [cmd]    command description for cmd
       -a, -i      adapter name or number
       -d          output format signed integer
       -u          output format unsigned integer
       -x          output format hexadecimal
       ver          get version information
       cmds         generate a short list of available commands

       ioctl_echo check ioctl functionality
       up       reinitialize and mark adapter up (operational)
       down     reset and mark adapter down (disabled)
       out      mark adapter down but do not reset hardware(disabled)
                On dual-band cards, cards must be band-locked before use.
   ```

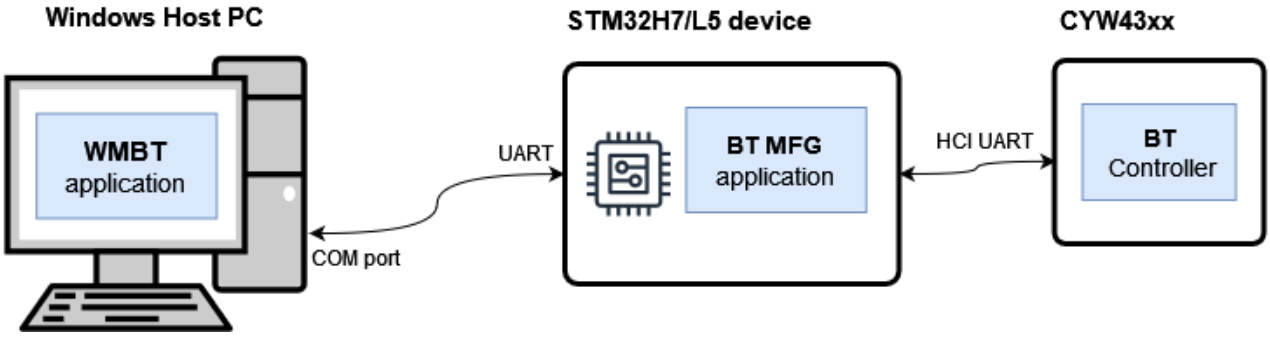
5.3 Bluetooth Manufacturing Test Application for FreeRTOS

The Bluetooth Manufacturing Test Application is used to validate the Bluetooth Firmware and RF performance of Cypress SoC Bluetooth BR/EDR/LE devices.

The Bluetooth MFG Application acts as a transport layer between the host "WMBT tool" and Bluetooth Firmware. Mfg Test Application receive commands from the WMBT tool and forwards them to the Bluetooth firmware. The Bluetooth MFG Application also relays the response received back from Bluetooth firmware.

This example demonstrates how an STM32H7 can be used to host CYW43xxx connectivity devices.

Test setup are shown below:



5.3.1 Hardware

Refer to the section on the STM32 hardware configuration descriptions as appropriate:

Using STM32H747 DISCO Kit

5.3.2 Other software

This application requires WMBT Tool running on a windows PC and uses UART port for communication with target. The pre-built executables for WMBT Tool are available in bt_mfg_tester/wmbt-tool-bin directory, which sync from btsdk-utils. and user guide is in Bluetooth Manufacturing Test Tool.

IQxel tool as transmitter to send fixed count test packet which to ensure whatever is sent from the transmitter would received by the receiver, without any error.

Using Sniffer to ensure whatever is test packet is in same transmit channel, packet length and data patterns from transmitter.
Better to test it in the shield room to avoid air interference

5.3.3 Project components

The following are the only components used in this project:

- Bluetooth/btstack
- Bluetooth/bluetooth-freertos
- Platform/pal (PAL, HAL, core-lib)
- Platform/abstraction-rtos (configured for the FreeRTOS kernel)

- Platform/device (configured as CYW43012)

5.3.4 Example project start/import

You can open the example by copying this example from the Pack to an appropriate location:

C:\Users\<USER>\STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\1.2.0\Projects\STM32H747I-DISCO\Applications\bt_mfg_tester

Once you have copied the example, you can then open it in STM32CubeMX and export to your IDE using the following steps from Example project start/import, Generate code, Build the project from Wi-Fi Scan example.

5.3.5 Project hardware setup

Refer to section Hardware Setup.

5.3.6 Operation

1. Go to WMBT tool directory
2. Reset the Board by pressing Reset button
3. Run the command on Windows Host for the proper BT Chip on target board.
4. Observe the output of the command

   List of wmbt commands with BLE function which can be retrieved by typing --help Partial output of the command and display is below.

   ```
   wmbt reset COMx
   ```

5.3.6.1 Example output

```
# wmbt.exe reset COM5

cmd resp: MBT_BAUD_RATE:  115200
TRANSPORT_MODE: 0 (HCI)

Opened COM5 at speed: 115200
Close Serial Bus
Opened COM5 at speed: 115200

Sending HCI Command:
0000 < 01 03 0C 00 >

Received HCI Event:
0000 < 04 0E 04 01 03 0C 00 >

Success
Close Serial Bus
```

6 Special options and setup

6.1 STM32H7xx – using serial flash

There may be a need for extra internal Flash space when running applications on STM32H7xx. A significant amount of internal Flash can be saved if Wi-Fi stack is placed on external Serial Flash memory module. The STM32H747I-DISCO board has MT25QL512ABB8ESF-0SIT memory IC present for this purpose.

- STM32H747I-DISCO has serial Flash in dual-bank Quad-SPI mode
- STM32H7 has QSPI HW block

Additional settings are needed to enable placing Wi-Fi stack firmware on external memory:

1. Linker script (*.ld) has external memory address defined:

```
QSPI    (rx)    : ORIGIN = 0x90000000,       LENGTH = 131072K
```

2. Linker script has section name defined where WiFi stack will be located during linkage:

```
.whd_fw :
{
__whd_fw_start = .;
KEEP(*(.whd_fw))
__whd_fw_end = .;
} > OSPI
```

3. Preprocessor macro name added:

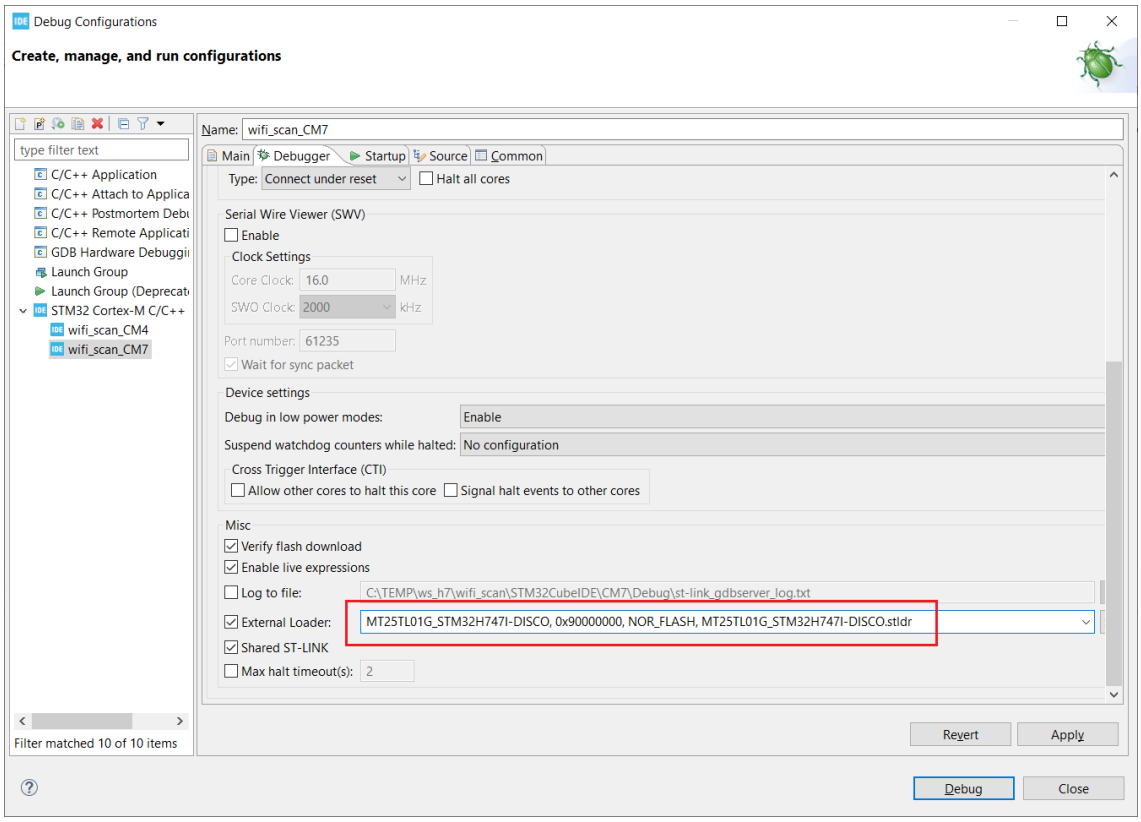
```
CY_STORAGE_WIFI_DATA=".whd_fw"

BSP-files have to be added:
BSP\stm32h747i_discovery_qspi.c
BSP\stm32h747i_discovery_qspi.h
BSP\Components\mt25tl01g\mt25tl01g.c(*.h)
BSP\Components\mt25tl01g\mt25tl01g.c(*.h)
BSP\Components\mt25tl01g\mt25tl01g_conf.h
```

4. BSP Initialization routine call have to be added:

```
/* Configure External Memory to Memory Mapped Mode*/
/* QSPI info structure */
BSP_QSPI_Info_t pQSPI_Info;
uint8_t status;
/*##-1- Configure the QSPI device ###########################################*/
/* QSPI device configuration */
BSP_QSPI_Init_t init ;
init.InterfaceMode=MT25TL01G_QPI_MODE;
init.TransferRate= MT25TL01G_DTR_TRANSFER ;
init.DualFlashMode= MT25TL01G_DUALFLASH_ENABLE;
status = BSP_QSPI_Init(0,&init);
if (status != BSP_ERROR_NONE)
{
     printf("\r\n    ERROR: BSP_QSPI_Init() failed \r\n");
     Error_Handler();
}
/*##-2- Read & check the QSPI info ##########################################*/
/* Initialize the structure */
pQSPI_Info.FlashSize          = (uint32_t)0x00;
pQSPI_Info.EraseSectorSize    = (uint32_t)0x00;
pQSPI_Info.EraseSectorsNumber = (uint32_t)0x00;
pQSPI_Info.ProgPageSize       = (uint32_t)0x00;
pQSPI_Info.ProgPagesNumber    = (uint32_t)0x00;
/* Read the QSPI memory info */
BSP_QSPI_GetInfo(0,&pQSPI_Info);
```

```
/*##-6-Memory Mapped Mode ##################################################*/
status = BSP_QSPI_EnableMemoryMappedMode(0);
if (status != BSP_ERROR_NONE)
{
     printf("\r\n    ERROR: BSP_QSPI_EnableMemoryMappedMode() failed \r\n");
     Error_Handler();
}
```

5. Programming of the Serial Flash should be performed with appropriate Flash Loader selection:



Note: External flash (MT25QL512ABB8ESF) requires 3.3 V for normal operation.

6.2 STM32H7xx – using internal flash (BANK2) to store Wi-Fi FW

The internal flash space on STM32H7xx is divided into two banks: BANK1 (1M) is used for CM7, BANK2 (1M) is used for CM4. The steps to reuse part of BANK2 to store Wi-Fi firmware images:

1. Update the linker script (*.ld):
   - Add WIFI_FLASH memory definition to the MEMORY section of the linker script:

```
WIFI_FLASH (rx) : ORIGIN = 0x08180000, LENGTH = 512K /* ORIGIN address of
BANK2 (0x08100000) with 512K offset */
```

   - Define the whd_fw section where the WiFi FW will be located during linkage:

```
.whd_fw :
{
__whd_fw_start = .;
KEEP(*(.whd_fw))
__whd_fw_end = .;
} > OSPI
```

2. Add the reprocessor macro name:

```
CY_STORAGE_WIFI_DATA=".whd_fw"
```

6.3 STM32L562 – using serial flash

The wifi application can't fit STM32L5x internal flash due to size constraints. MCU has 512kB of area when connectivity firmware reaches over 1MB.

To resolve this external memory module is used, present on STM32L562E-DK board.

The project has following additional settings made to enable placing WiFi stack firmware on external memory:

1. Linker script (*.ld) has external memory address defined:

```
OSPI(rx)    : ORIGIN = 0x90000000,       LENGTH = 65536K
```

2. Linker script has section name defined where WiFi stack will be located during linkage:

```
.whd_fw :
{
__whd_fw_start = .;
KEEP(*(.whd_fw))
__whd_fw_end = .;
} > OSPI
```

3. Preprocessor macro name added:

```
CY_STORAGE_WIFI_DATA=".whd_fw"
```

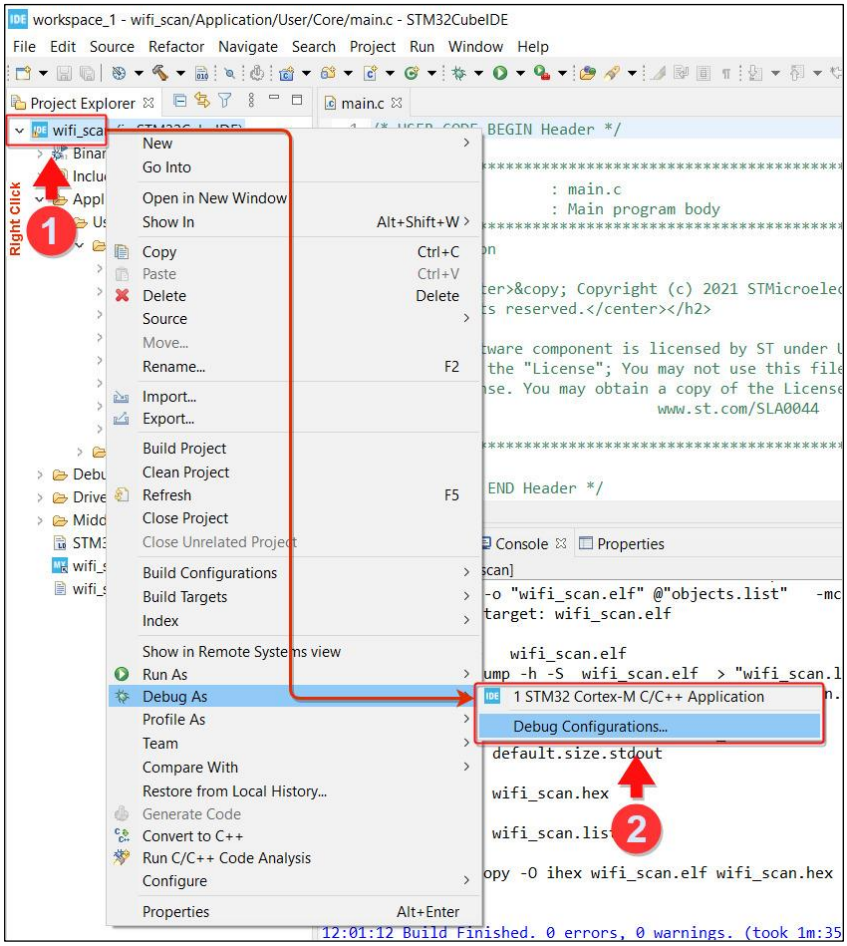
With given setup compiler and linker will split a resulting image into two pieces, which will reside in both – internal and external memory of an STM32L562E-DK.

6.4 STM32L562 – serial flash programming

6.4.1 Using STM32CubeMX IDE

To program resulting image into the target device an appropriate Flash loader has to be selected:

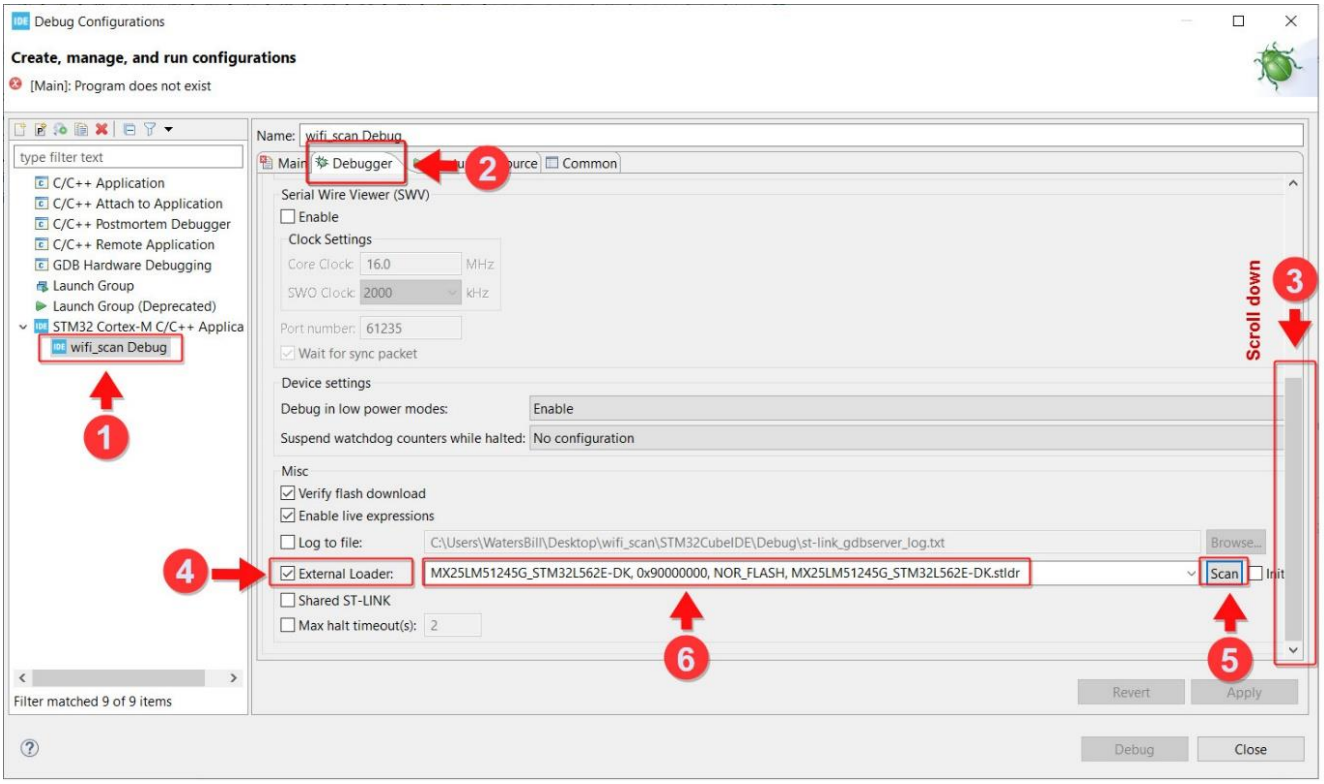
1. Right-click on a project, select **Debug As > Debug Configurations...**



2. Select the external loader (see steps illustrated in the following image).

   - Select “wifi_scan Debug” and select the **Debugger** tab.
   - Scroll down and select the **External Loader** check box.
   - Click **Scan** to refresh the list of available Flash loaders.
   - Select the appropriate loader:

     MX25LM51245G_STM32L562E-DK, 0x90000000, NOR_FLASH, MX25LM51245G_STM32L562E-DK.stldr



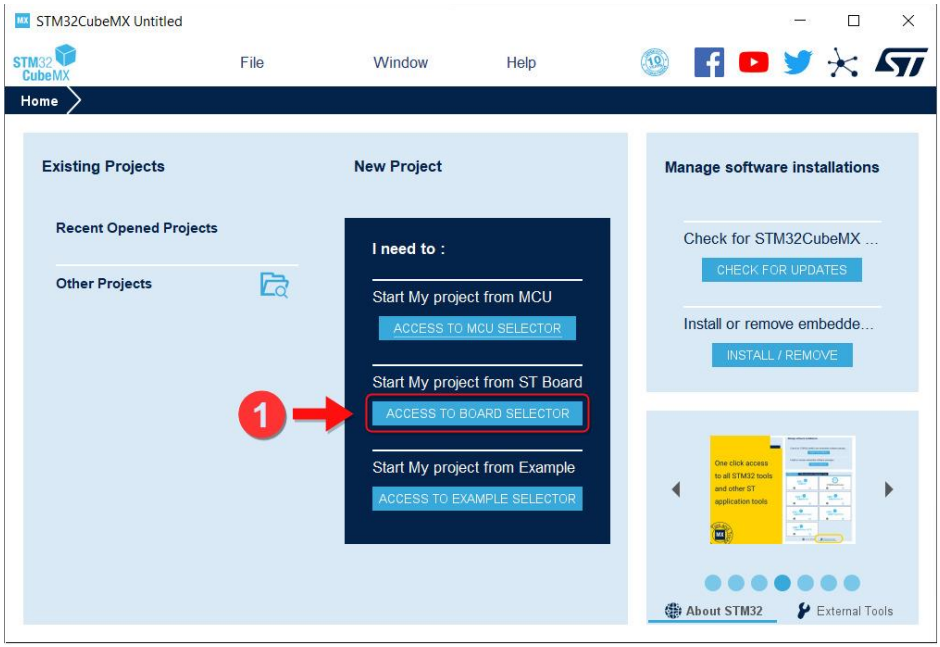
3. Program your target with “Run” or “Debug” command.

7 Create a new project from scratch

This section takes you step-by-step through the process of creating a project file from scratch.

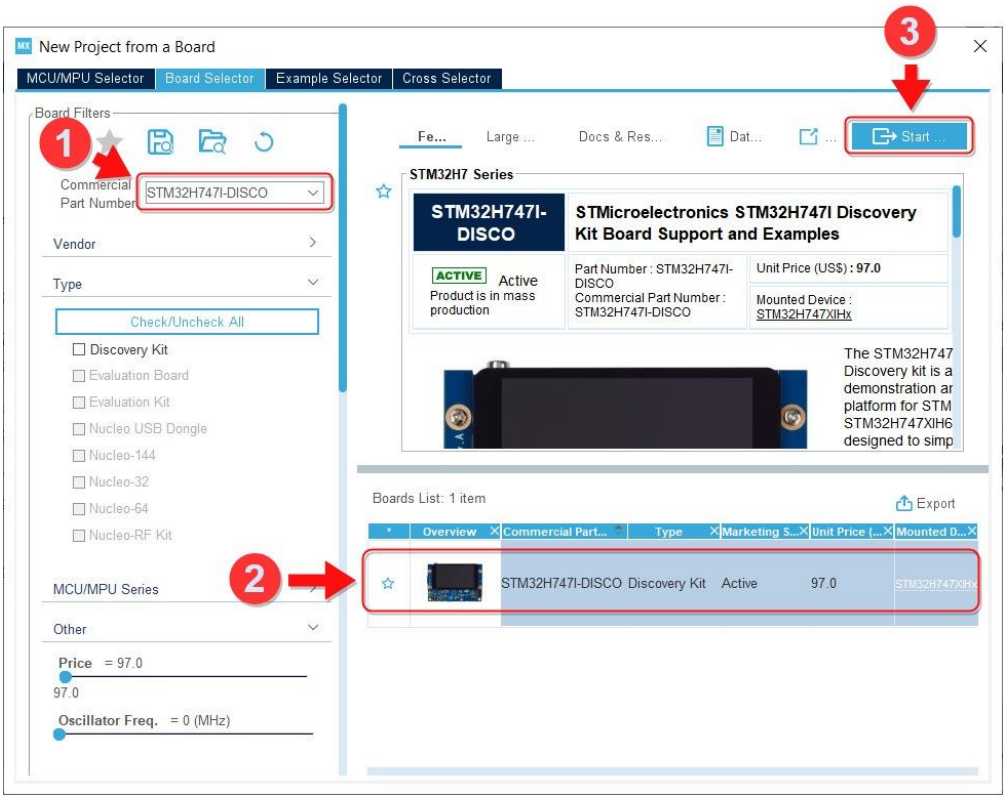
7.1 Create a project for specific board

1. Start creating a project via the Access to Board Selector option.



2. Select a board.

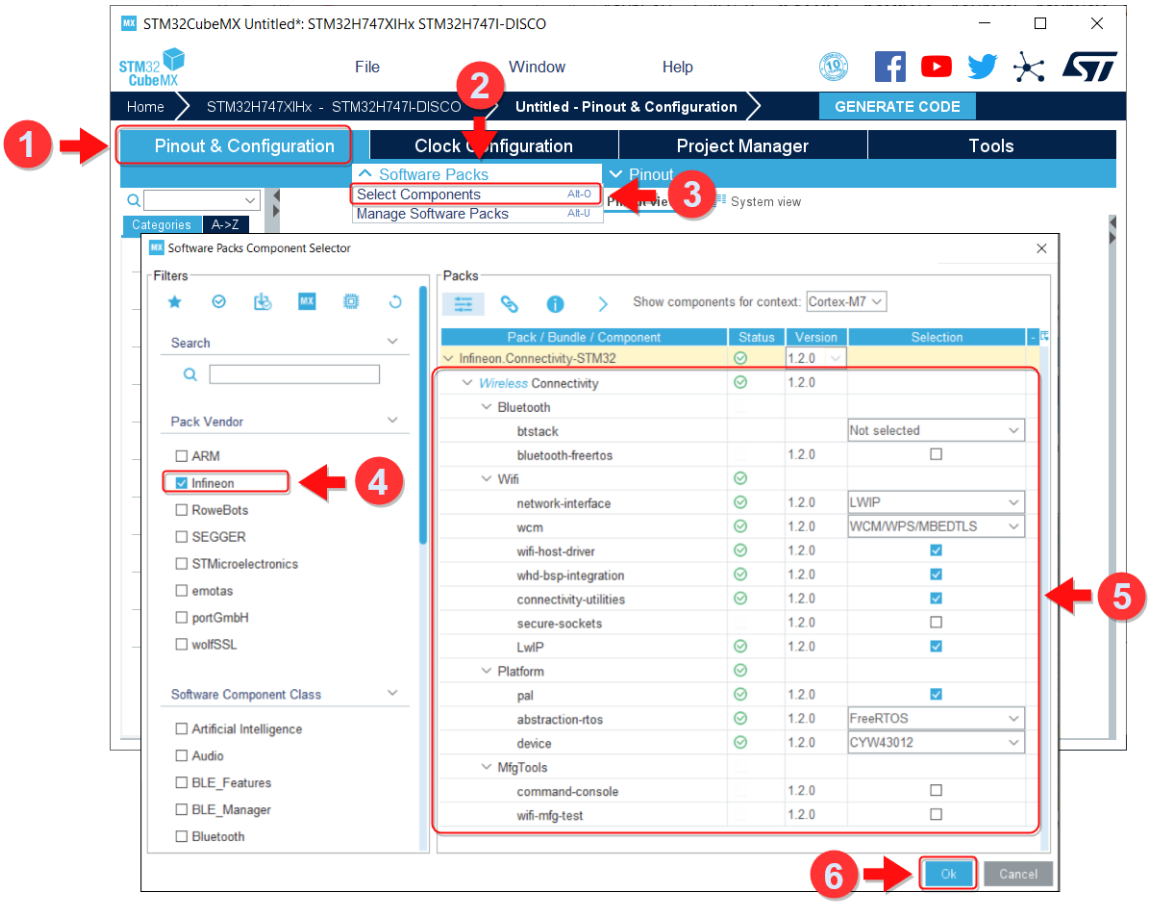
   - Enter/select the board number and click on your selected board.
   - Select **Start Project**.



7.2 Enable software components from STM32 connectivity expansion pack

1. Select the **Pinout & Configuration** tab.
2. Select **Software Packs > Select Components**.

   This will show a list of the installed packs and their contents.

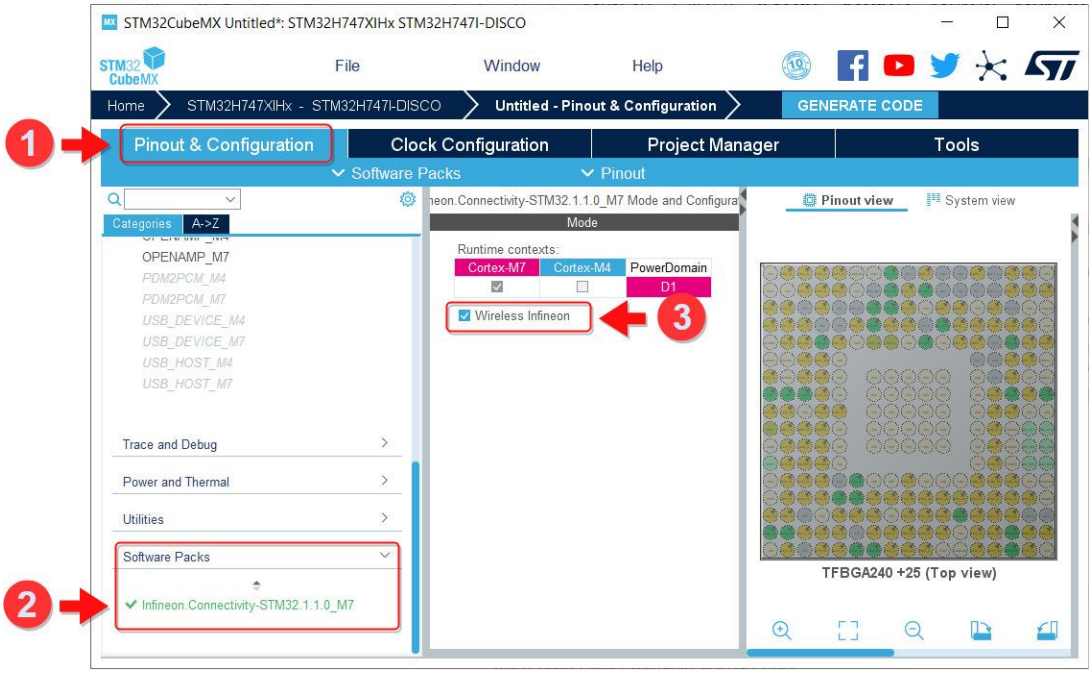


3. Select the components you need for your project. All projects will use three ‘Platform’ components. If you are using Wi-Fi, select all the ‘Wifi’ components. If you are using Bluetooth LE, select all ‘Bluetooth’ components.

Note: Platform components are required for each type of application – either Wi-Fi-only, Bluetooth-only or combined.

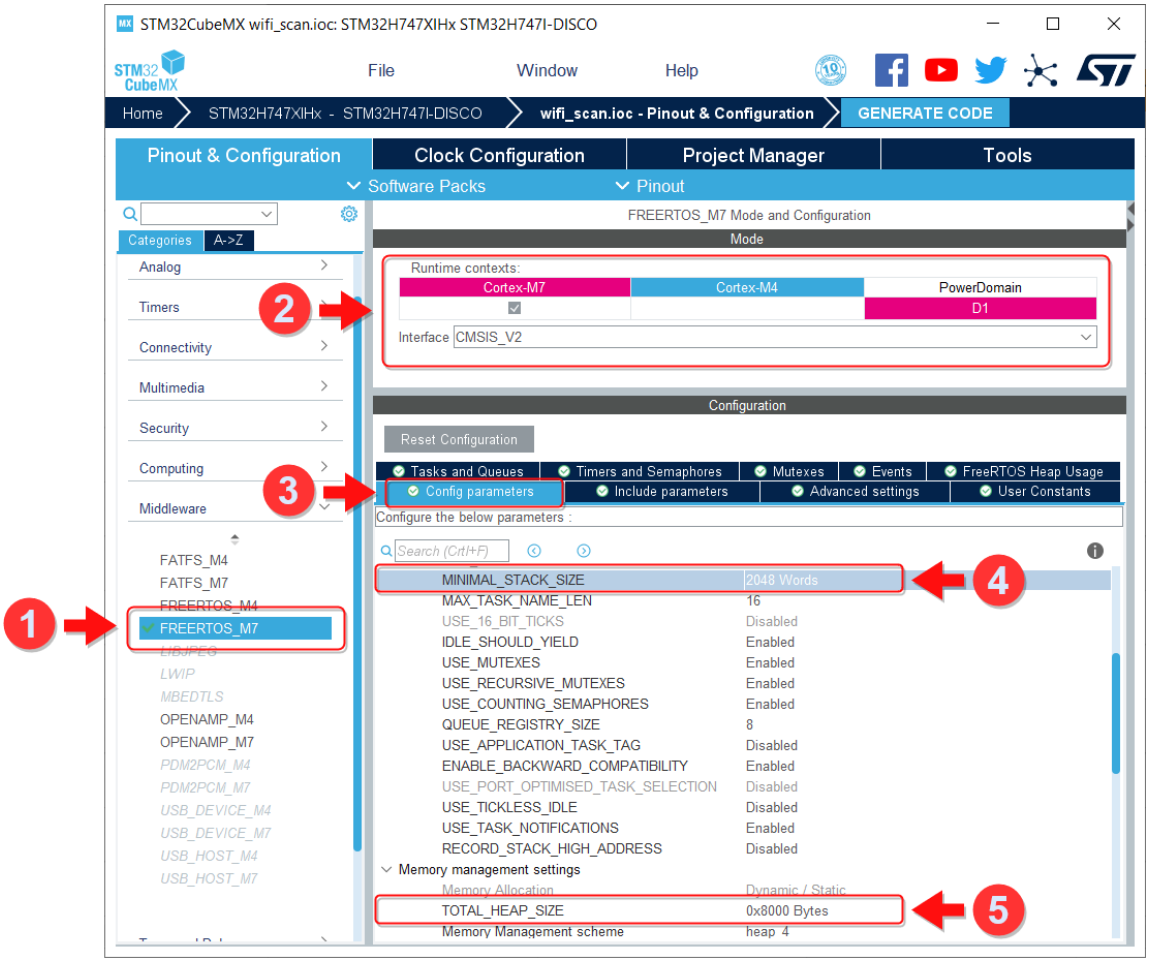
- For the ‘Platform / device’ component, select the appropriate connectivity device for your system (CYW43012, CYW4343W or CYW43438, etc).
- For the ‘Wifi / network-interface’ component, select the appropriate network interface for your system (LwIP or NetxDuo).
- For the ‘Wifi / wcm’ component, select the appropriate variant (WCM or WCM /WPS/MBEDTLS).
  - WCM Variant compiles only Wi-Fi connection manager files which provides set of APIs that can be used to establish and monitor Wi-Fi connections on Infineon platforms that support Wi-Fi connectivity
  - WCM /WPS/MBEDTLS Variant also includes APIs to connect to a Wi-Fi network using Wi-Fi Protected Setup (WPS) methods which uses MBED TLS security stack.

4. Enable Software pack in the **Pinout & Configuration** tab.

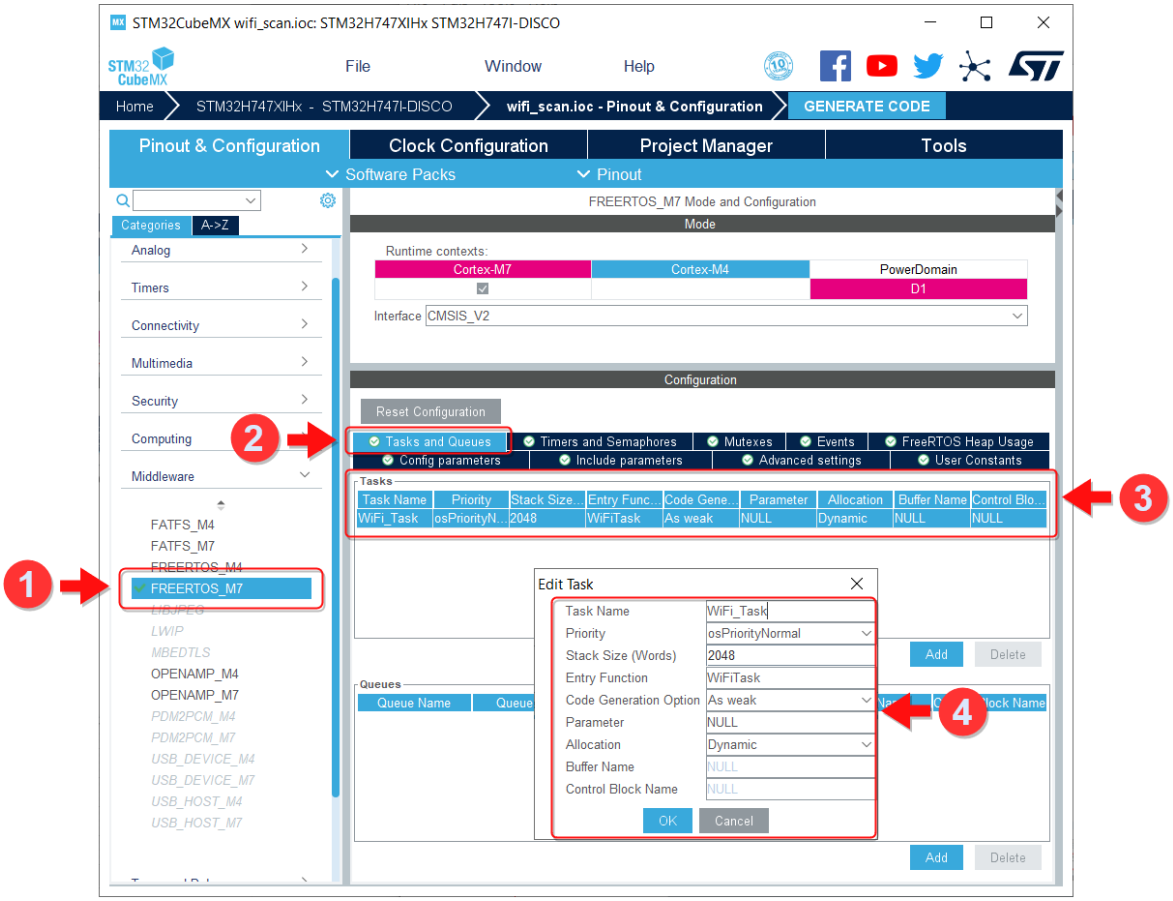


7.3 FreeRTOS configuration

1. Select FreeRTOS version and configure Stack Size and Heap size as required for the application.



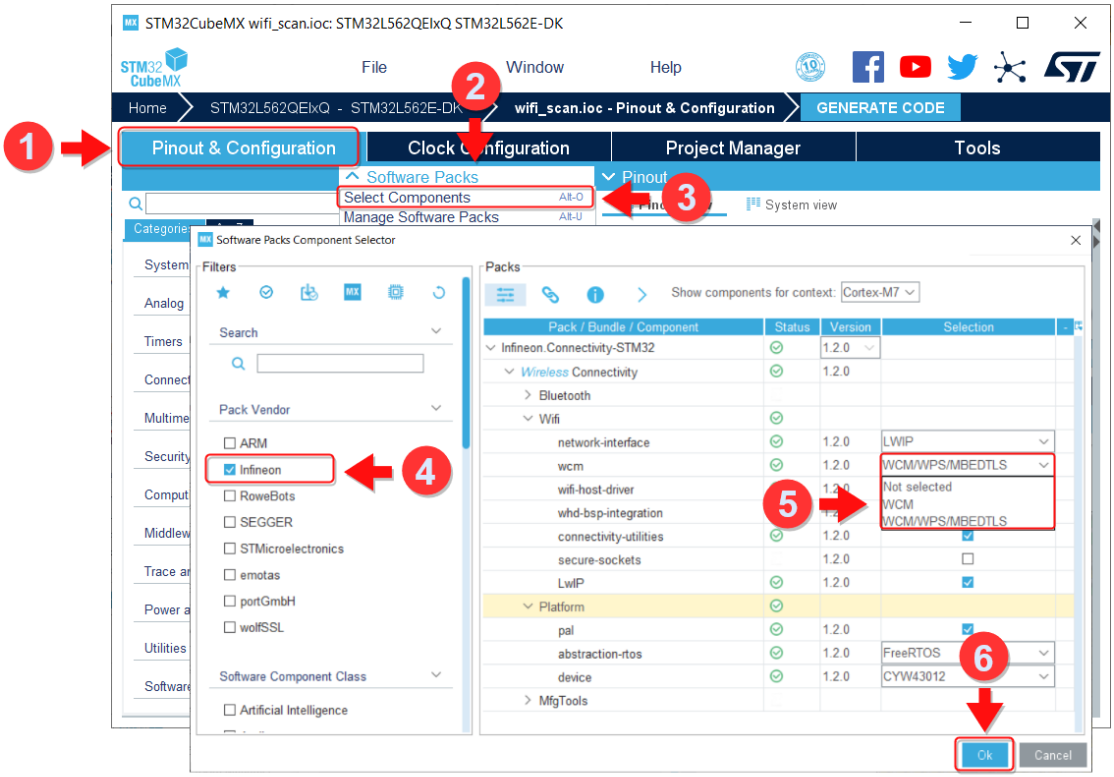
2. Configure Default task and its stack size.



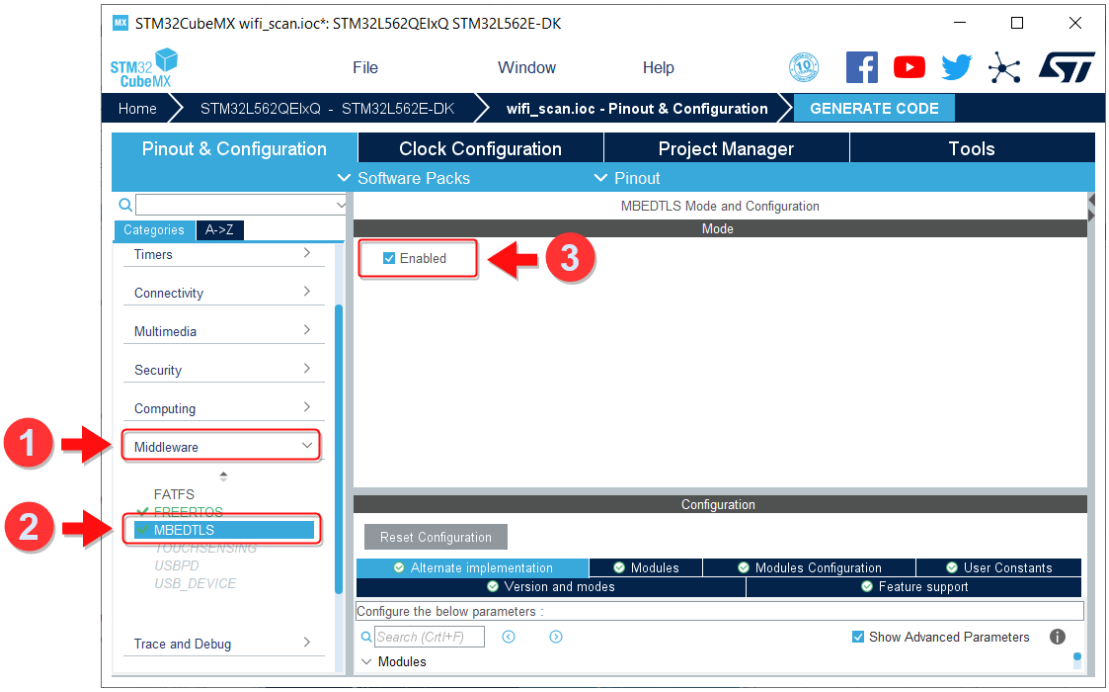
7.4 MbedTLS configuration

The mbedTLS is required by LwIP (Lightweight IP) WCM (WiFi Connection Manager) Pack's components. To enable mbedTLS. The STM32L5 MCU is used as an example to demostrate Crypto features (AES, HASH, etc) HW acceleration configuration:

1. Open the project's *.IOC file w/ STM32CubeMx.
2. Navigate to Infineon Pack's components and switch WCM to WCM/WPS/mbedTLS.



3. Navigate to **Select Components**, select **Middleware** and then select **MBEDTLS** for target device and select the **Enabled** check box.



4. Ensure that the following features and modes are enabled:

   MBEDTLS_ENTROPY_HARDWARE_ALT
   MBEDTLS_AES_ROM_TABLES
   MBEDTLS_CIPHER_MODE_CBC
   MBEDTLS_NO_PLATFORM_ENTROPY
   MBEDTLS_ENTROPY_FORCE_SHA256
   MBEDTLS_AES_C
   MBEDTLS_SHA256_C

Note: Set "Not defined" for unneeded modes to reduce memory consumption and eliminate unused code.

By performing those steps:

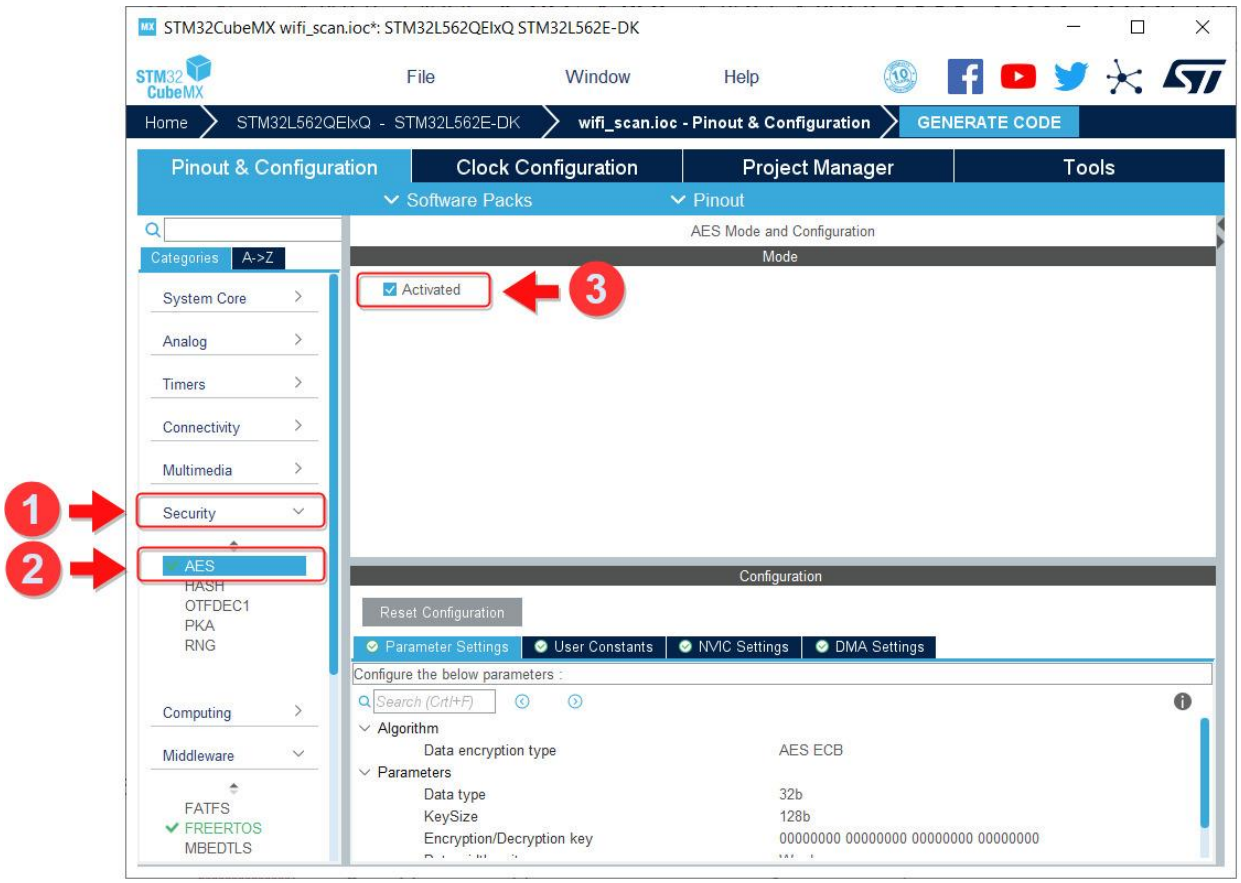
- mbedTLS sources are added to application
- mbedTLS config is applied to support Infineon's connectivity middleware

7.4.1 Crypto HW acceleration

STM32 offers HW acceleration for the following crypto-related functions:

	STM32H7	STM32L5	Notes
RNG	+	+	
AES		+	AES-128/256 (ECB, CBC, CTR, GCM GMAC, CCM)
HASH		+	SHA1, SHA224, SHA256, MD5 HMAC SHA1, HMAC SHA224, HMAC SHA256, HMAC MD5
PKA		+	Public Key Cryptography
OTFDEC1		+	On-the-fly decryption of Octo-SPI external memories (AES-128)

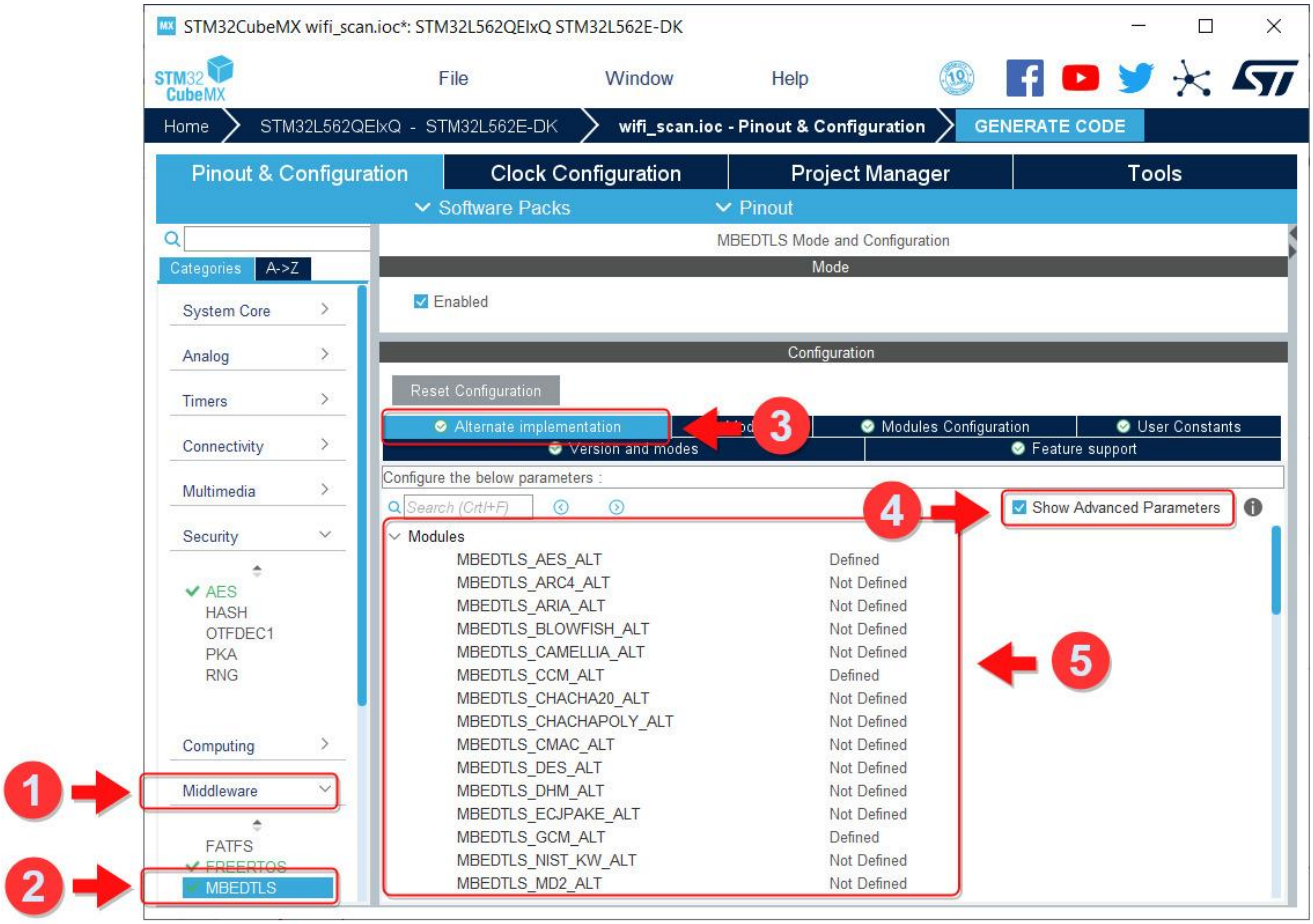
The IP modules listed above must be enabled (Activated) from the "Security" section of STM32CubeMX configurator.



Create a new project from scratch

To enable HW acceleration the following literals have to be defined for mbedTLS (should be done in STM32CUbeMX configurator):

MBEDTLS_AES_ALT
MBEDTLS_CCM_ALT
MBEDTLS_GCM_ALT
MBEDTLS_MD5_ALT
MBEDTLS_SHA1_ALT
MBEDTLS_SHA256_ALT
MBEDTLS_ENTROPY_HARDWARE_ALT

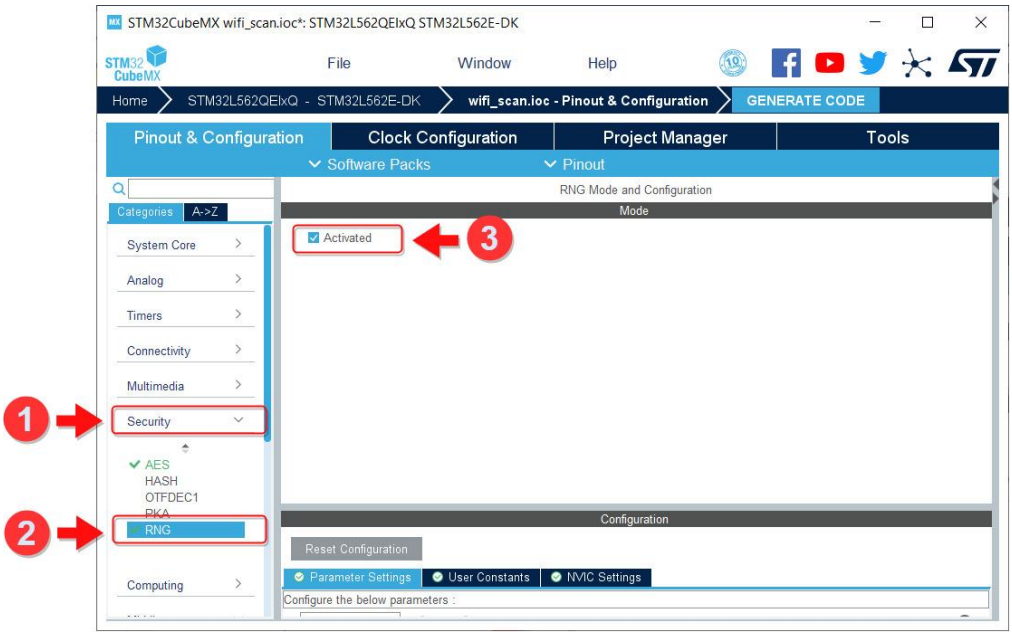


After these steps, source files marked with *_alt suffixes (meaning "alternative", not the original mbedTLS version) will be added into the user's project. They will provide an interface between thembedTLS crypto functions and its HAL HW counterpart.

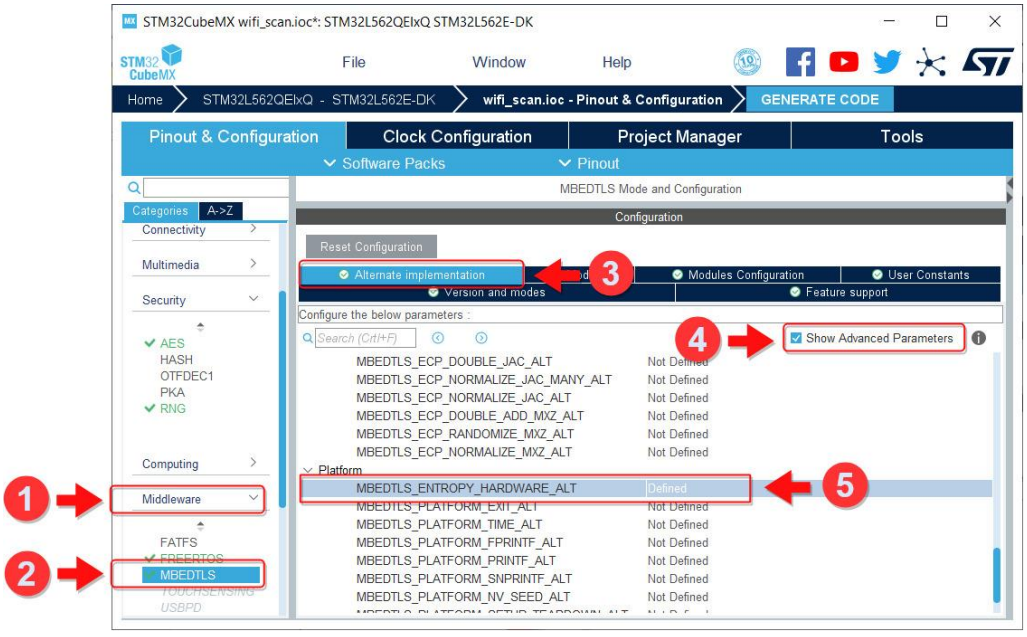
7.4.2 HW source of entropy example

To obtain a good source of entropy used for a public/private key generation and other cryptographic functions:

1. Enable the RNG module in the STM32CubeMX configurator.



2. Set MBEDTLS_ENTROPY_HARDWARE_ALT to "Defined" in the STM32CubeMX configurator:



3. The tool will add hardware_rng.c source file to the user's project.
4. This will provide the mbedtls_hardware_poll() implementation, which relies on the devices' HW RNG IP block.
5. A call to the standard STM32 HAL RNG API (HAL_RNG_GenerateRandomNumber()) will be used by the system to fulfill the mbedTLS entropy pool.

7.5 Configure resources for WIFI connectivity

The following Peripherals and I/O lines are required for the host MCU to communicate to Infineon connectivity device(s) for Wi-Fi:

7.5.1 SDIO

SDIO is used as an interface with Infineon Connectivity devices.

The SDMMC HAL component is required for STM32 host MCU to access/control Infineon connectivity device(s).

1. Add the API call at initialization with appropriate handle passed in:

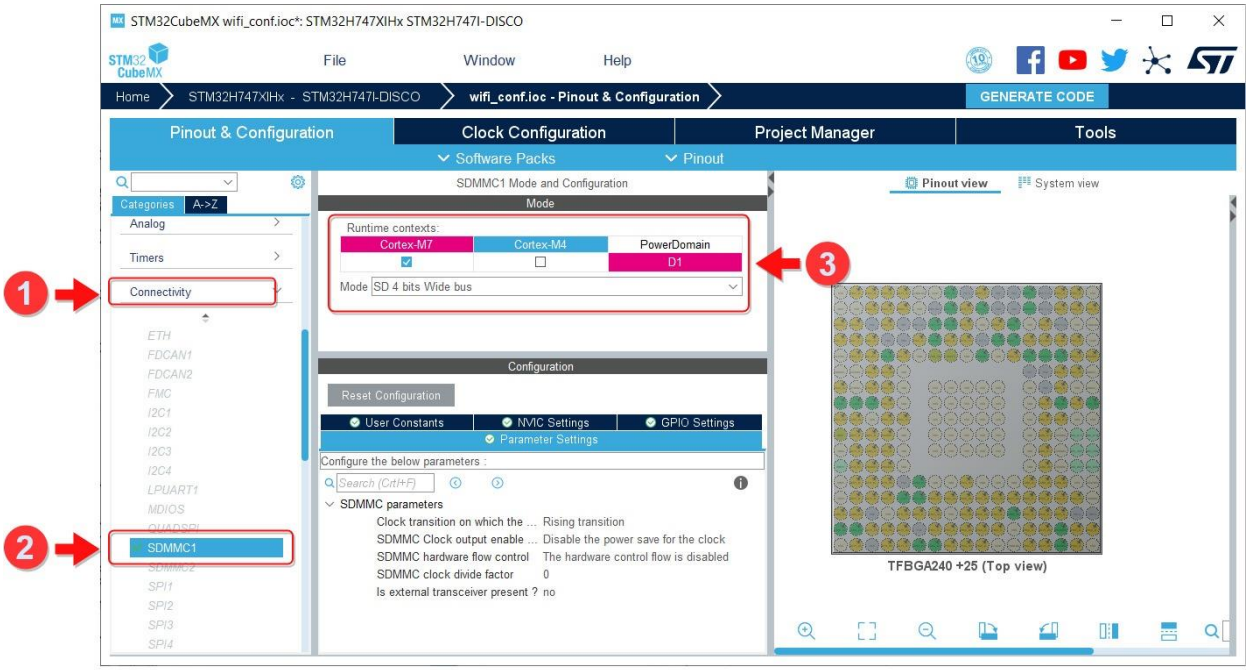
```
SD_HandleTypeDef SDHandle = { .Instance = SDMMC1 };
cy_rslt_t result = stm32_cypal_wifi_sdio_init(&SDHandle);
```

2. SDMMC Interrupt handler must be overwriting in application and call stm32_cyhal_sdio_irq_handler function:

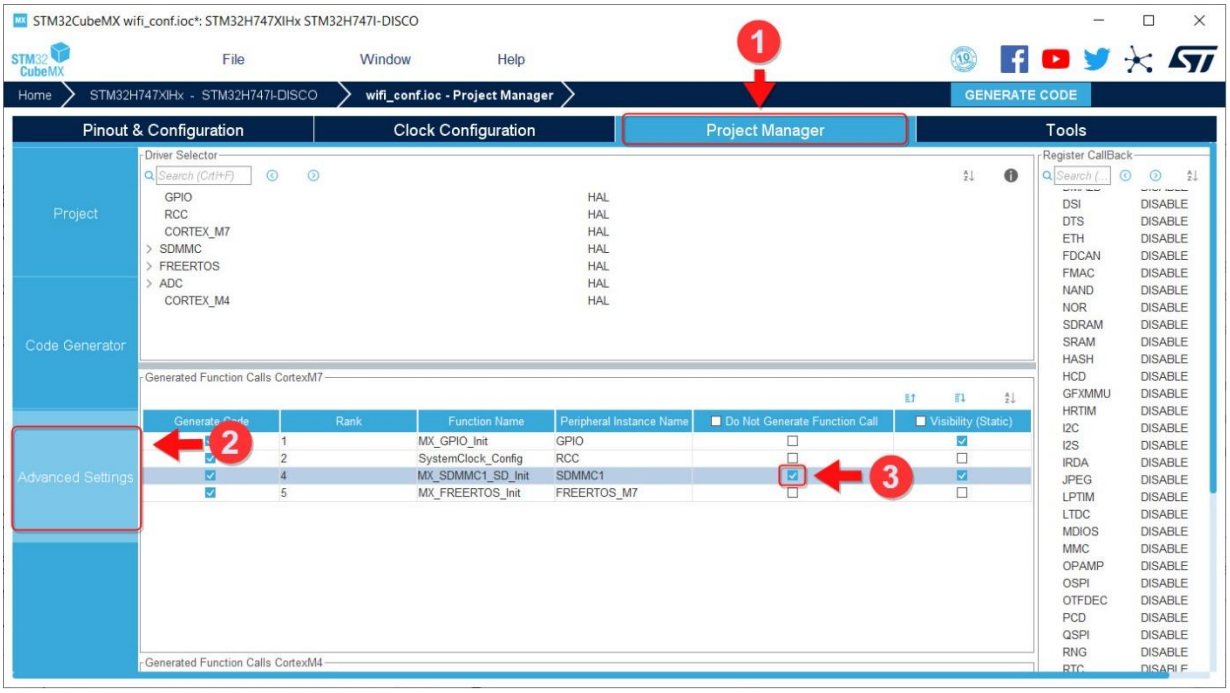
```
void SDMMC1_IRQHandler(void)
{
    stm32_cyhal_sdio_irq_handler();
}
```

Make sure the SDMMC instance selected has its pins routed to the Infineon Connectivity device. Follow the steps listed to enable/configure SDIO in STM32CubeMX:

1. Enable SDMMC block in **STM32CubeMX > Pinout & Configuration > Connectivity**.



2. Disable generation function call of SDMMC initialization (MX_SDMMC_SD_Init).



7.5.2 Control pins

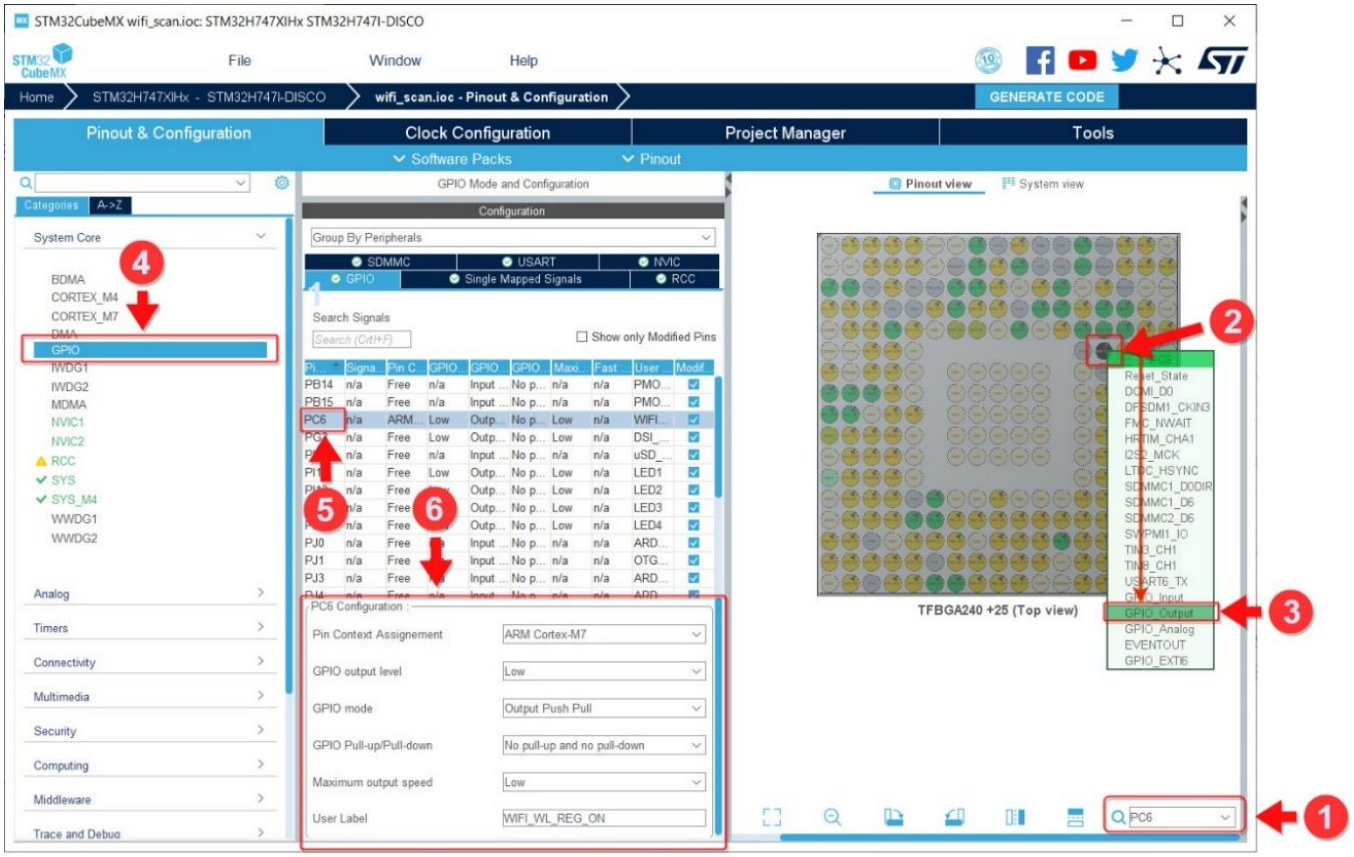
Infineon Connectivity devices require control lines to be connected to host MCU:

Line Name	FW Name	Description
WL_REG_ON	WIFI_WL_REG_ON	This is a power pin that shuts down the device WLAN section.
WL_HOST_WAKE	CYBSP_WIFI_HOST_WAKE	WLAN Host Wake: Active Low (OOB IRQ)
WL_DEV_WAKE		WLAN Device Wake *Note: WL_DEV_WAKE is not used in current version of PAL.*

7.5.2.1 WL_REG_ON

A power pin that shuts down the device WLAN section. WL_REG_ON must be configured as output with following parameters:

GPIO Parameter	Value	Note
Direction	GPIO_Output	
Pin Context Assignment	ARM Cortex-M7	Assign to core, where Connectivity run.
GPIO output level	Low	
GPIO mode	Output Push Pull (PP)	
GPIO Pull-up/Pull-down	No pull-up and no pull-down	
Maximum output speed	Low	
User label	WIFI_WL_REG_ON	



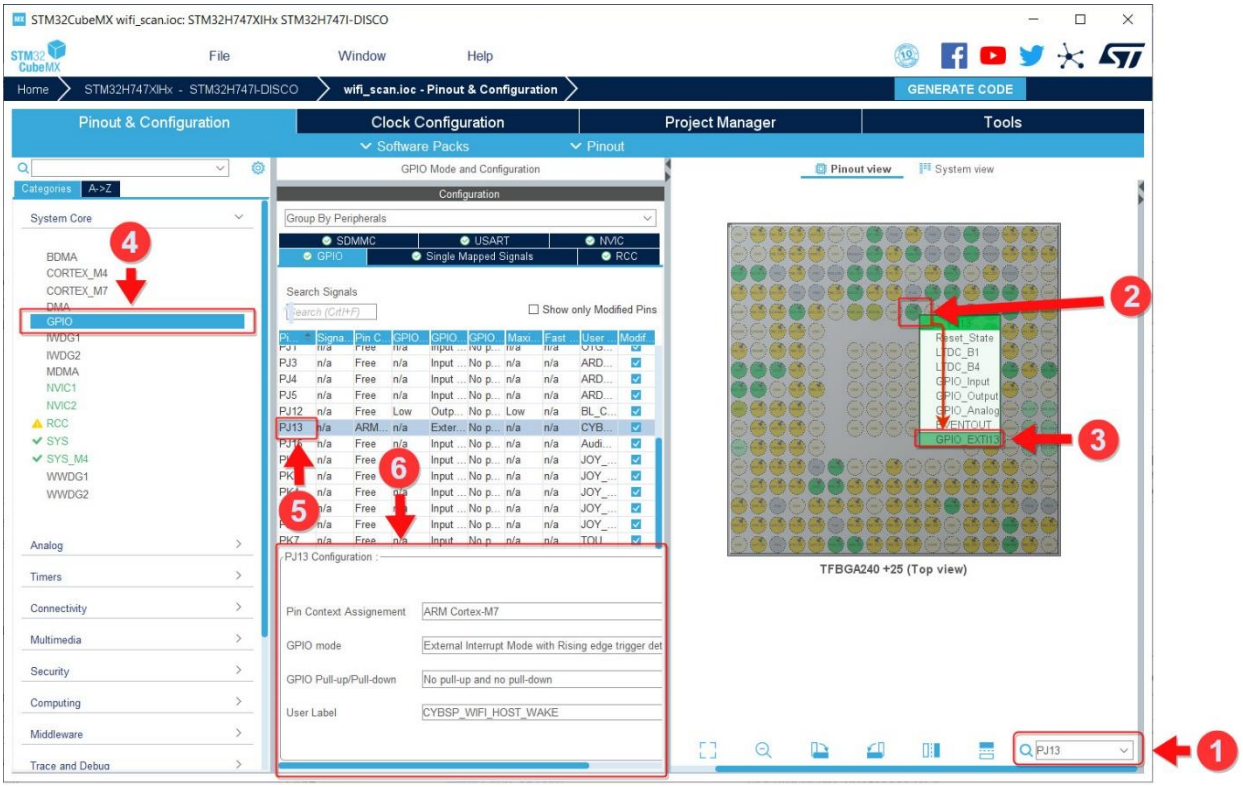
7.5.2.2 WL_HOST_WAKE

Host MCU Wake signal from WLAN section. WL_HOST_WAKE must be configured in External Interrupt mode / EXTI with following parameters:

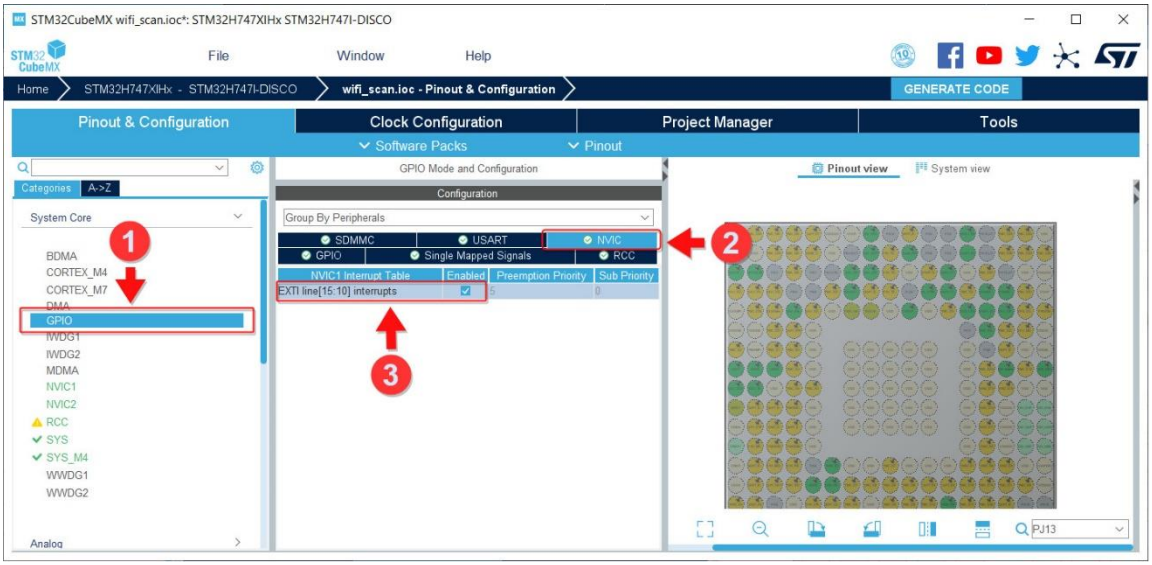
GPIO Parameter	Value	Note
Direction	GPIO_EXTIxx	
Pin Context Assignment	ARM Cortex-M7	Assign to core, where Connectivity runs.
GPIO mode	External Interrupt mode with Rising edge trigger detection	
GPIO Pull-up/Pull-down	No pull-up and no pull-down	
User label	CYBSP_WIFI_HOST_WAKE	
NVIC for EXTI	Enable	

Create a new project from scratch

1. Configure in STM32CubeMX:



2. Enable NVIC interrupt for EXTI line:



3. EXTI Callback handler must be overwriting in application and call stm32_cyhal_gpio_irq_handler function:

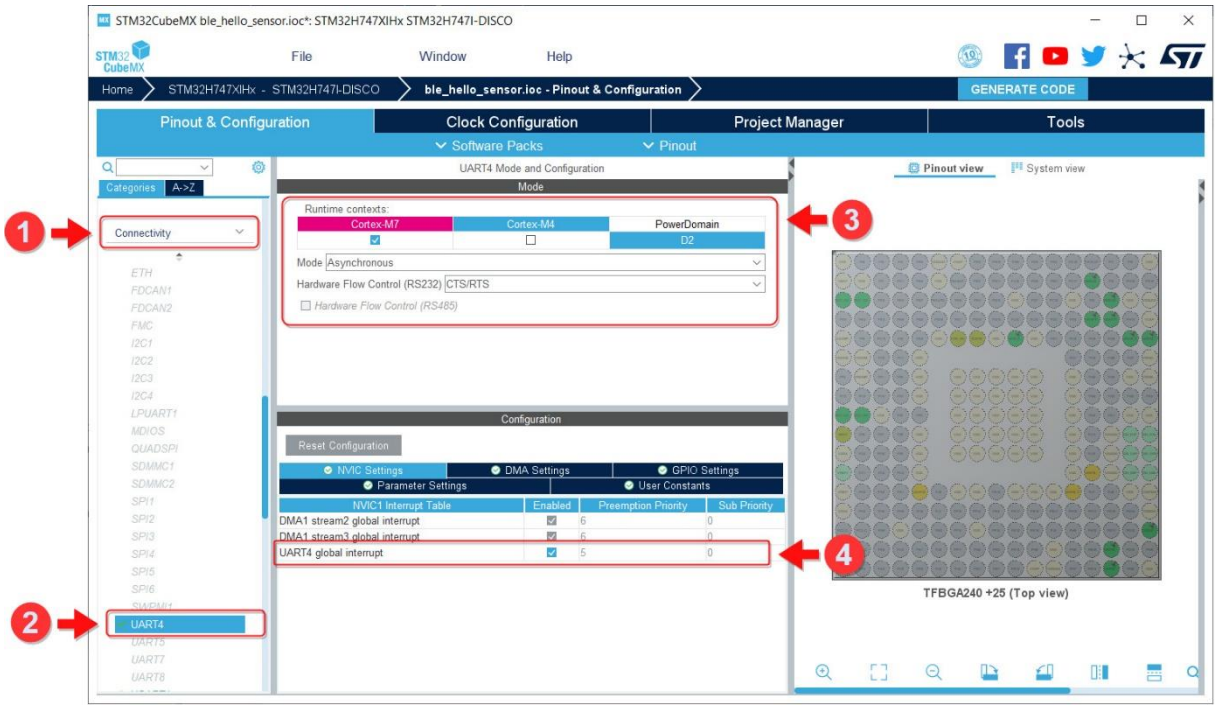
```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    stm32_cyhal_gpio_irq_handler(GPIO_Pin);
}
```

7.6 Configure resources for Bluetooth connectivity

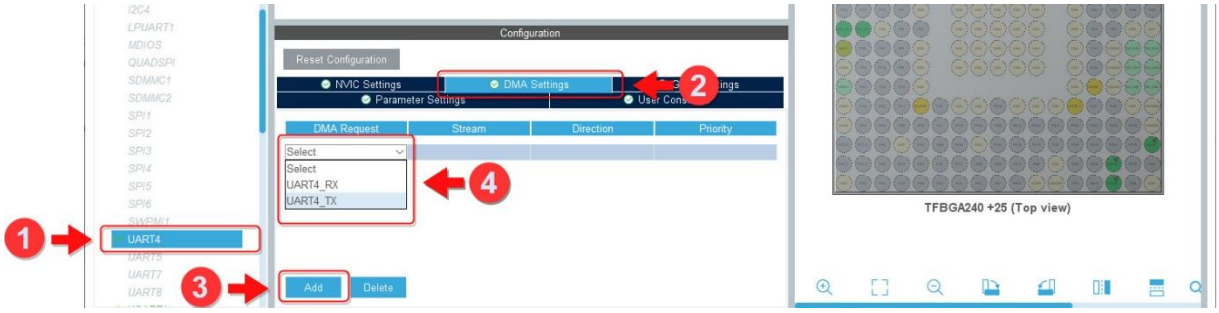
The following Peripherals and I/O lines required for the host MCU to communicate to Infineon connectivity device(s) for Bluetooth:

7.6.1 UART

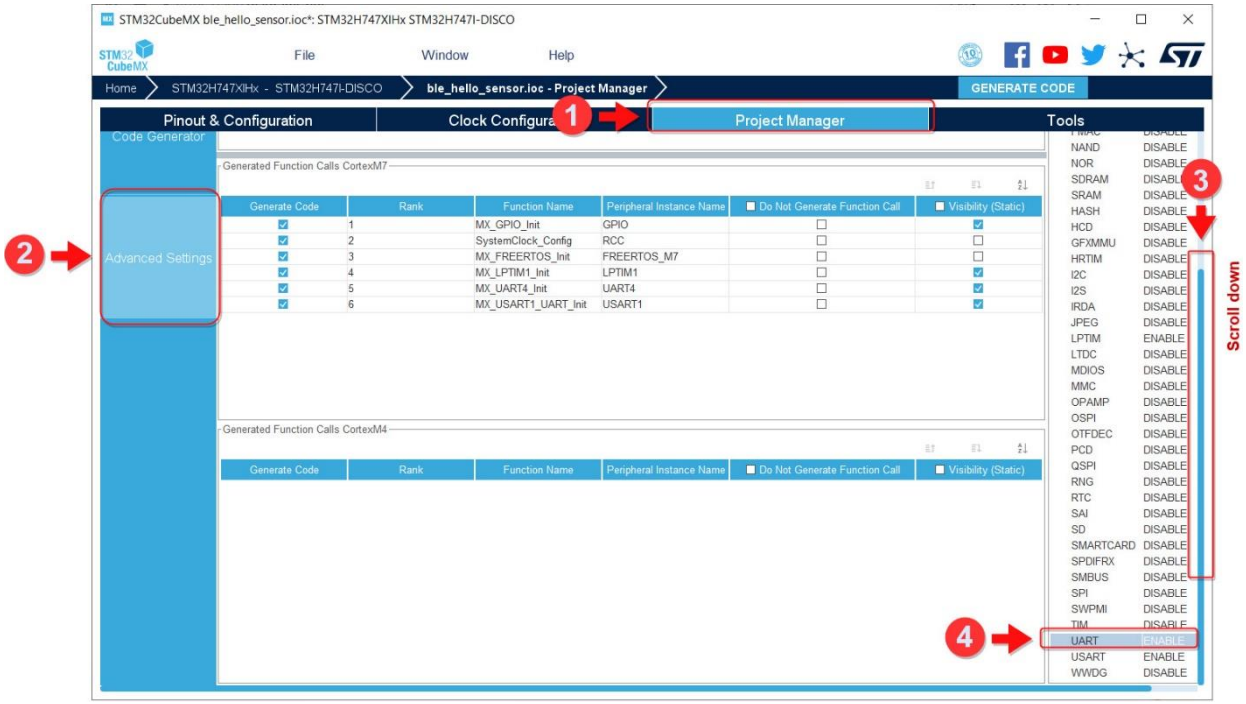
1. Enable UART block in **STM32CubeMX > Pinout & Configuration > Connectivity.**
2. Configure Mode as **Asynchonous**.
3. Configure Hardware Flow Control (RS232) as **CTS/RTS.**
4. Enable UART interrupt in **NVIC Settings**.



5. Add DMA for RX and TX in **DMA Settings**. Use default settings for RX/TX.

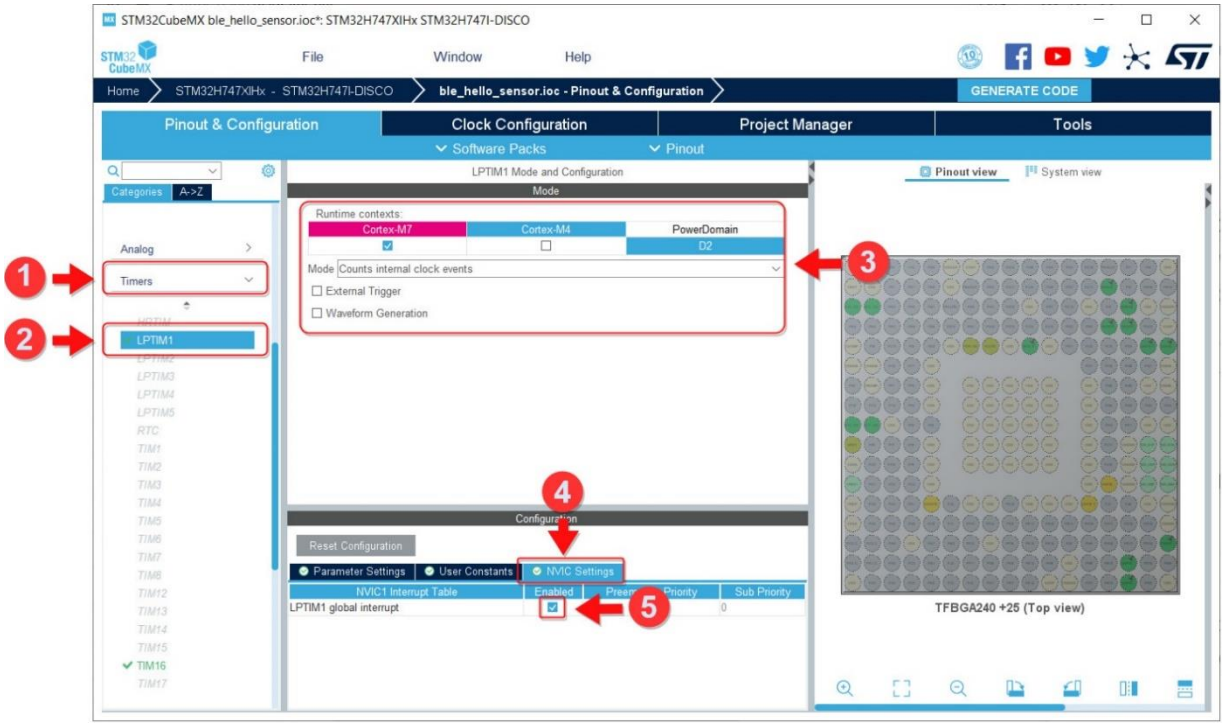


6. Enable UART Callback.

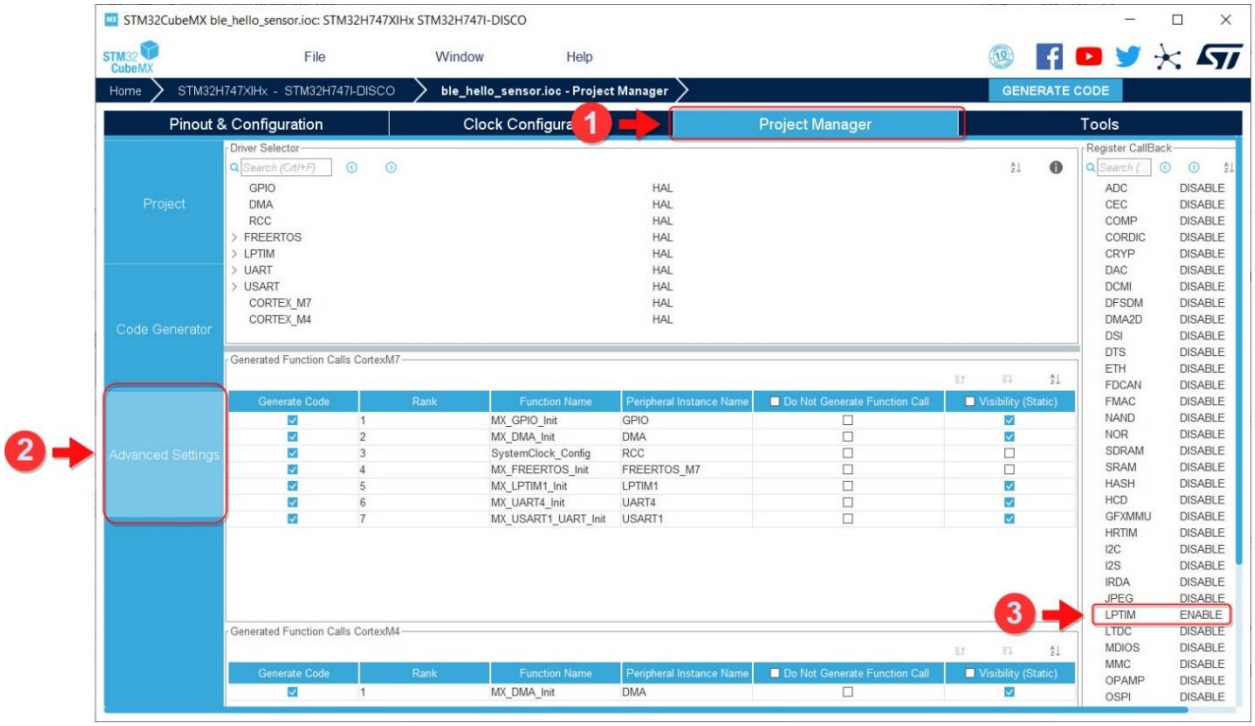


7.6.2 LPTIMER

1. Enable LPTIMER block in **STM32CubeMX > Pinout & Configuration > Timers**.
2. Configure Mode as **Counts internal clock events**.
3. Enable LPTIMER interrupt in **NVIC Settings**.



4. Enable LPTIM Callback.



7.6.3 Control pins

Infineon Connectivity devices require control lines to be connected to host MCU:

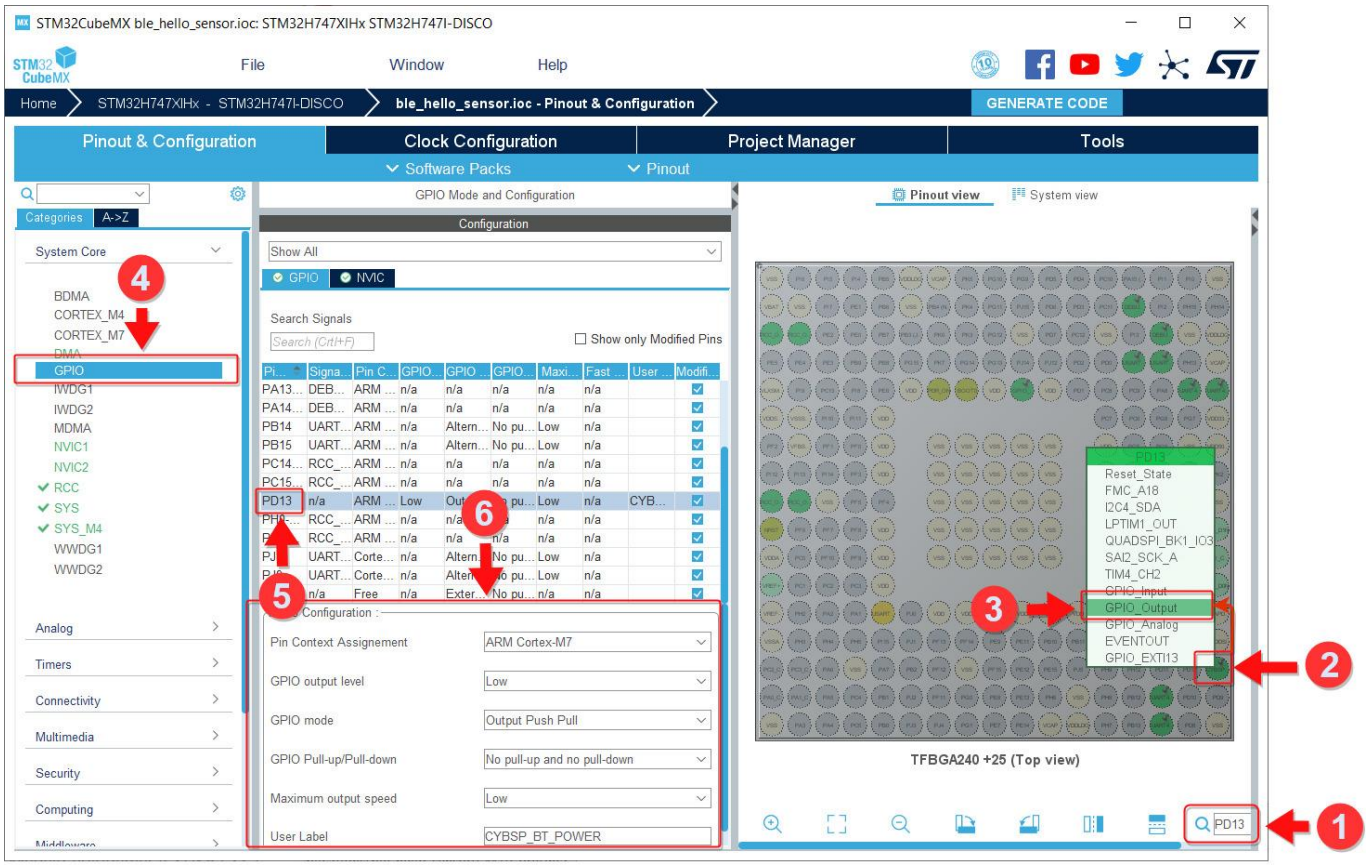
Line Name	FW Name	Description
BT_REG_ON	CYBSP_BT_POWER	Used by the PMU to power-up or power-down the internal regulators used by the Bluetooth section.
BT_HOST_WAKE	CYBSP_BT_HOST_WAKE	Bluetooth device wake-up: Signal from the host to the CYW43xx indicating that the host requires attention. • Asserted: The Bluetooth device must wake-up or remain awake. • Deasserted: The Bluetooth device may sleep when sleep criteria are met. The polarity of this signal is software configurable and can be asserted HIGH or LOW. *Note: BT_HOST_WAKE is not used in current version of PAL.*
BT_DEV_WAKE	CYBSP_BT_DEVICE_WAKE	Host wake-up. Signal from the CYW43xx to the host indicating that the CYW43xx requires attention. • Asserted: host device must wake-up or remain awake. • Deasserted: host device may sleep when sleep criteria are met. The polarity of this signal is software configurable and can beasserted HIGH or LOW *Note: BT_DEV_WAKE is not used in current version of PAL.*

7.6.3.1 BT_REG_ON

A power pin that shuts down the device Bluetooth section. BT_REG_ON must be configured as output with the following parameters:

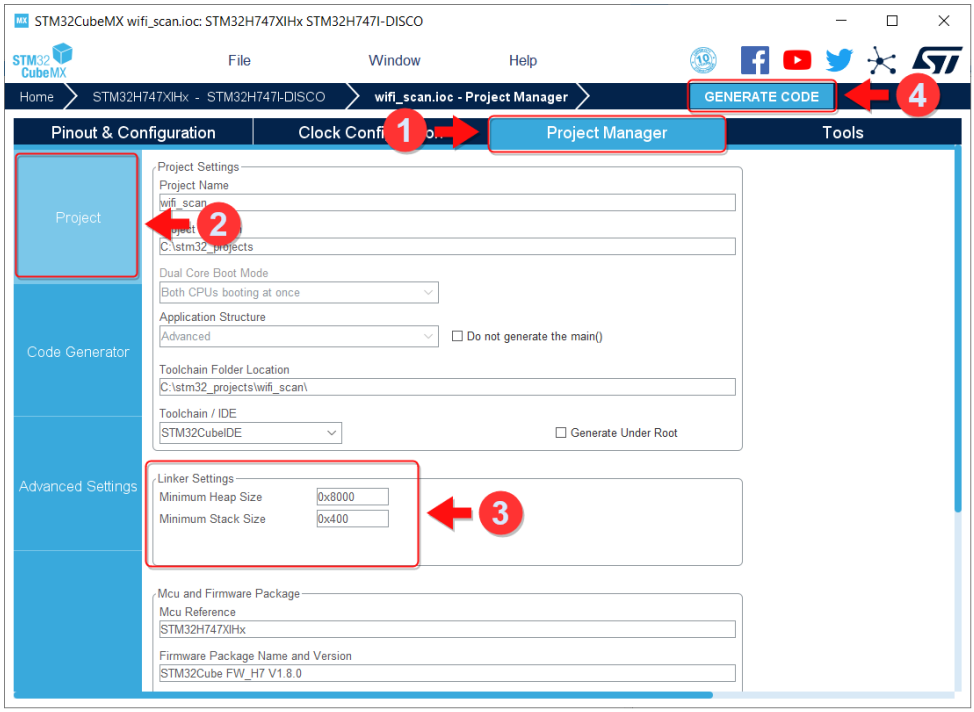
GPIO Parameter	Value	Note
Direction	GPIO_Output	
Pin Context Assignment	ARM Cortex-M7	Assign to core, where Connectivity run.
GPIO output level	Low	
GPIO mode	Output Push Pull (PP)	
GPIO Pull-up/Pull-down	No pull-up and no pull-down	
Maximum output speed	Low	
User label	CYBSP_BT_POWER	

Configuration in STM32CubeMX:



7.7 Heap and stack configuration

Configure Heap and Stack size required for the example app.



7.8 Generating code

1. After clicking **Generate Code**, copy the following files from existing examples provided along with the pack:

   - cybsp.h
   - lwipopts.h

   Location of these files in the pack:

   STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\1.2.0\Projects\STM32H747I-DISCO\Applications\wifi_scan\Core\Inc

2. Add the following to the *FreeRTOSConfig.h* file:

```
/* Enable using CY_HAL for rtos-abstraction */
#define CY_USING_HAL
```

3. Update the following fields in the *cybsp.h* file to match the configurations done in the Configuring Control pins section

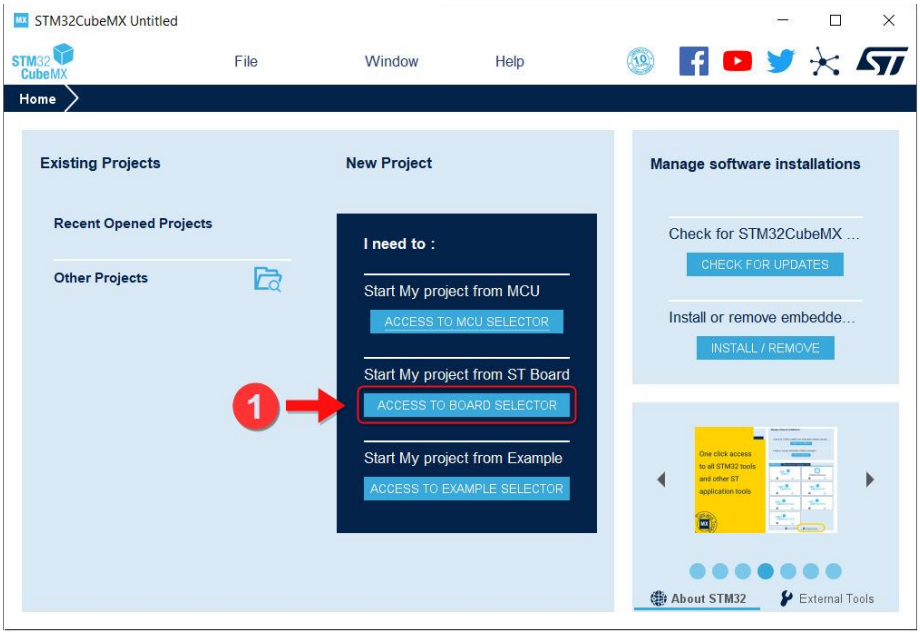
```
/** These names are explicitly referenced in the support libraries */
#define  CYBSP_WIFI_WL_REG_ON           ***
#define  CYBSP_WIFI_HOST_WAKE           ***
```

8 Create a new project for non-H7 MCU boards

This section explains how to create new example project for any non-H7 MCU boards using the expansion pack.

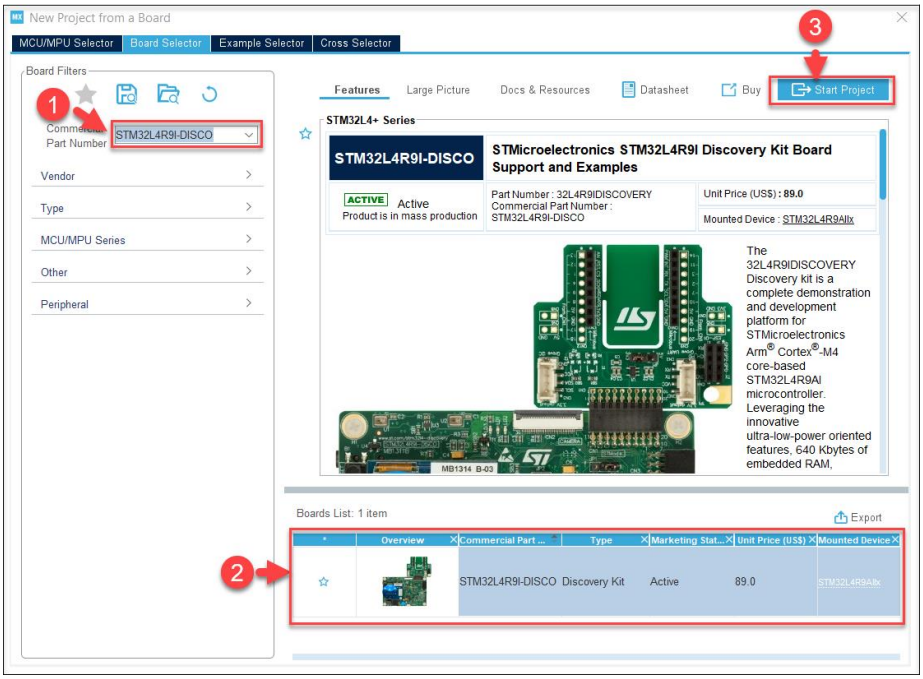
8.1 Creating a project

1. Start creating a project via the Access to Board Selector option.



2. Select a board like STM32L4R9I-DISCO

   - Enter/select the board number (STM32L4R9I-DISCO) and click on your selected board
   - Select Start Project.



3. Select Software Components from STM32 Connectivity Expansion Pack

   - Select the **Pinout & Configuration** tab.
   - Select **Software Packs > Select Components**. This will show a list of the installed packs and their contents.
   - Platform/device is selected as CYW43438 for reference along with other components required for the Wi-Fi Example.
   - Enable Software components as required for the Wi-Fi Example.
   - Refer to Enable Software components from STM32 Connectivity Expansion Pack.

8.2 FreeRTOS configuration

Follow same steps as mentioned in FreeRTOS Configuration.

8.3 Other configurations

1. Configure SDMMC (refer to SDIO).
2. Configure Control Pins (refer to Control Pins).
3. Configure Heap and Stack size (refer to Heap and Stack Configuration).

8.4 Changes required in PAL library

By default, Expansion pack supports only H7 MCU variant. The following changes are required to support other MCU variants.

1. stm32_cyhal_common.h
   (Middlewares\Third_Party\Infineon_Wireless_Infineon\pal\targets\TARGET_STM32\Inc) folder

```
#elif defined (STM32L4R9xx)
    #define TARGET_STM32L4xx
#elif defined (TARGET_STM32L4xx)
    #include "stm32l4xx.h"
    #include "stm32l4xx_hal.h"
    #include "stm32l4xx_hal_def.h"
```

2. stm32_cyhal_sdio_ex.h

- Define STM32_RCC_PERIPHCLK_SDMMC based in the SDMMC* type supported by MCU variant.
- For L4, it is RCC_PERIPHCLK_SDMMC1:

```
#elif defined (TARGET_STM32L4xx)
  /* RCC clock for SDMMC */
  #define STM32_RCC_PERIPHCLK_SDMMC RCC_PERIPHCLK_SDMMC1
```

3. stm32_cyhal_gpio.c

   Define “exti_table” based on the IRQn_Type defined in the stm32l4r9xx.h.

8.5 Changes required in main.c

To enable SDMMC to work with Wi-Fi connectivity device:

1. The API call has to be added at initialization with appropriate handle passed in:

```
SD_HandleTypeDef SDHandle = { .Instance = SDMMC1 };
cy_rslt_t result = stm32_cypal_wifi_sdio_init(&SDHandle);
```

2. SDMMC Interrupt handler must be overwriting in application and call stm32_cyhal_sdio_irq_handler function:

```
void SDMMC1_IRQHandler (void)
{
    stm32_cyhal_sdio_irq_handler();
}
```

3. GPIO Interrupt handler must be overwriting in application and call stm32_cyhal_gpio_irq_handler function

```
void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
{
    stm32_cyhal_gpio_irq_handler (GPIO_Pin);
}
```

8.6 DMA configuration

PAL Library is currently supporting SDIO CMD53 transfer using Internal DMA Registers in SDMMC. If the MCU variant does not support IDMABASE, Use DMA Channels and Modify below functions to handle SDIO Command 53.

- cyhal_sdio_bulk_transfer
- stm32_cyhal_sdio_irq_handler

8.7 OctoSPI configuration

STM32L4R9I-DISCO has external flash memory available and can be used for placing the Wi-Fi Firmware.

1. Linker script (*.ld) change to address external memory:

```
OSPI(rx)    : ORIGIN = 0x90000000,       LENGTH = 131072K
```

2. Add Linker script with section name defining where WiFi Firmware needs to be placed:

```
.whd_fw :
{
__whd_fw_start = .;
KEEP(*(.whd_fw))
__whd_fw_end = .;
} > OSPI
```

3. Add Preprocessor macro name:

```
CY_STORAGE_WIFI_DATA=".whd_fw"
```

9 Known issues/limitations

This section lists the known issues/limitations of this release:

Problem	Component	Workaround
The Wifi-host-driver does not pass initialization on Laird Sterling LWB5+ M.2 connectivity module and returns the error: `"Timeout while waiting for function 2 to be ready"`	wifi-host-driver (nvram, CYW4373, STERLING-LWB5plus)	This issue will be fixed in the next release.
Fails to join to the Wi-Fi Protected Access 3 (WPA3) network.	CYW43012, wifi-host-driver, WPA3	For WHD v2.3.0 and later, the CYW43012 part needs an external supplicant provided by the host MCU in order to support the WPA3 Personal security. STM32CubeMx Release v1.2.0 has WHD v2.4.0, but does not provide the external supplicant for the STM32 host MCU. This issue will be fixed in the next release.
Sometimes, STM32 detects UART "Frame error" during the Bluetooth® LE communication (with CYW43012), which causes the Bluetooth® LE functionality to stop.	Bluetooth-freertos (CYW43012 BT FW)	Register a User UART Error Callback (by using HAL_UART_RegisterCallback function) with implementing the Bluetooth® LE or System reset.
STM32CubeIDE returns the linkage error "undefined reference to _nx_nd_cache***" when IPv6 is enabled in the NetxDuo configuration.	STM32CubeMx/ STM32CubeIDE	Manually add nx_nd_cache_***.c files from the MCU pack (e.g STM32Cube_FW_U5_V1.1.1\Middlewares\ ST\netxduo\common\src) to the project workspace.
STM32CubeMx does not remove sources/includes of the PDSC component from the project workspace (STM32CubeIde/EWARM), when another variant of this component is disabled or changed. It causes a build error when two versions of one component are added to the project (e.g. device CYW43012 and CYW4373)	STM32CubeMx/ STM32CubeIDE	Option 1: Manually remove files/includes of the previous component variant from the project workspace. Option 2: Remove the project workspace folder and generate a project from STM32CubeMx again. Be careful with the custom linker script – it may be missing after removing the project folder.

Revision history

Revision	Date	Description
**	03/25/2021	Initial release.
*A	11/04/2022	Updated from version 1.1.0 to version 1.2.0.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-04-01
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-32903

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.