Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

STM32 Connectivity Expansion Pack User Guide

Version 1.0.1

Overview

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.cypress.com/products/modustoolbox. 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.

Expansion Pack Contents

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

Component Name	Version	Details
wifi-host-driver	1.93.0	The WHD (Wi-Fi Host Driver) is an independent, embedded Wi-Fi Host Driver that provides a set of APIs to interact with Cypress 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	2.0.1	WCM (Wi-Fi Connection Manager) 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 Cypress platforms that support Wi-Fi connectivity.
wifi-mw-core	3.0.0	This Wi-Fi Middleware Core Library comprises core components needed for Wi-Fi connectivity support. The library bundles FreeRTOS, lwIP TCP/IP stack, and mbed TLS for security, Wi-Fi Host Driver (WHD), Secure Sockets interface, configuration files, and associated code to bind these components together.
whd-bsp-integration	1.1.1	WHD (WiFi Host Driver Board) 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.
connectivity-utilities	3.0.1	The connectivity utilities library is a collection of general purpose middleware utilities such as: JSON parser, Linked list, String utilities, Network helpers, Logging functions, Middleware Error codes. Several connectivity middleware libraries shall depend on this utilities library.
core-lib	1.1.4	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.
abstraction-rtos	1.3.0	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 the provide just enough support to allow for RTOS independent drivers and middleware.
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.

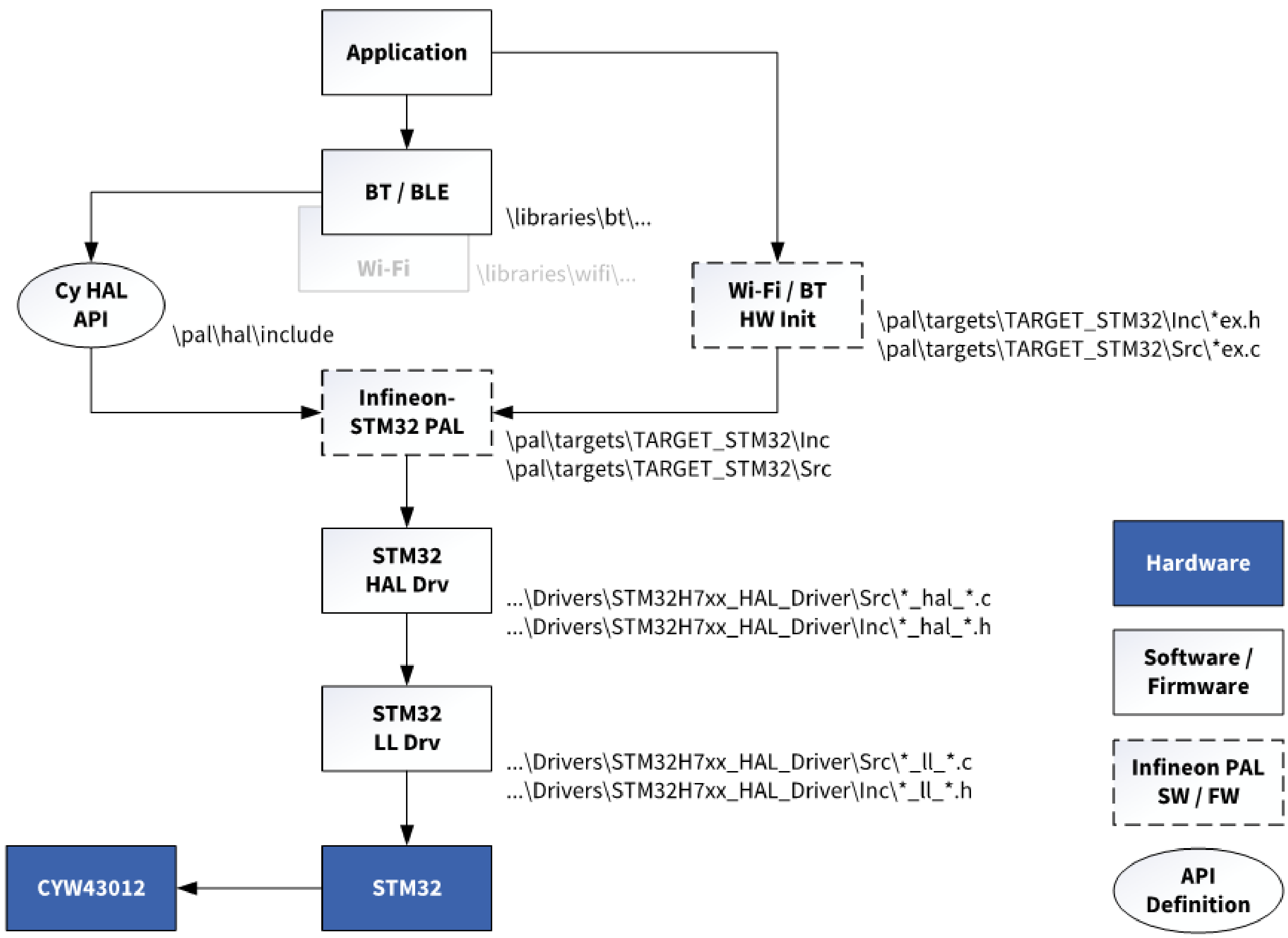
Component Name	Version	Details
pal	1.0.1	Infineon-STM32 Platform Adaptation Layer (PAL)
device	1.0.1	Selects appropriate CYW43xxx firmware and drivers for selected connectivity device.

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	1.8.0

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

Supported STM32 MCUs

- STM32H7xx

Supported STM32 Boards

- STM32H747I-DISCO Discovery kit

Supported Connectivity Modules

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

- CYW43012
- CYW4343W / CYW43438
- CYW4373

Compatible Software

- STM32 CubeMX 6.1.1
- STM32 CubeIDE 1.5.1
- IAR EWARM 8.50.4

Downloading the Expansion Pack

Download the expansion pack from GitHub in the Cypress Semiconductor organization:

https://github.com/cypresssemiconductorco/stm32-connectivity/releases/tag/release-v1.0.1

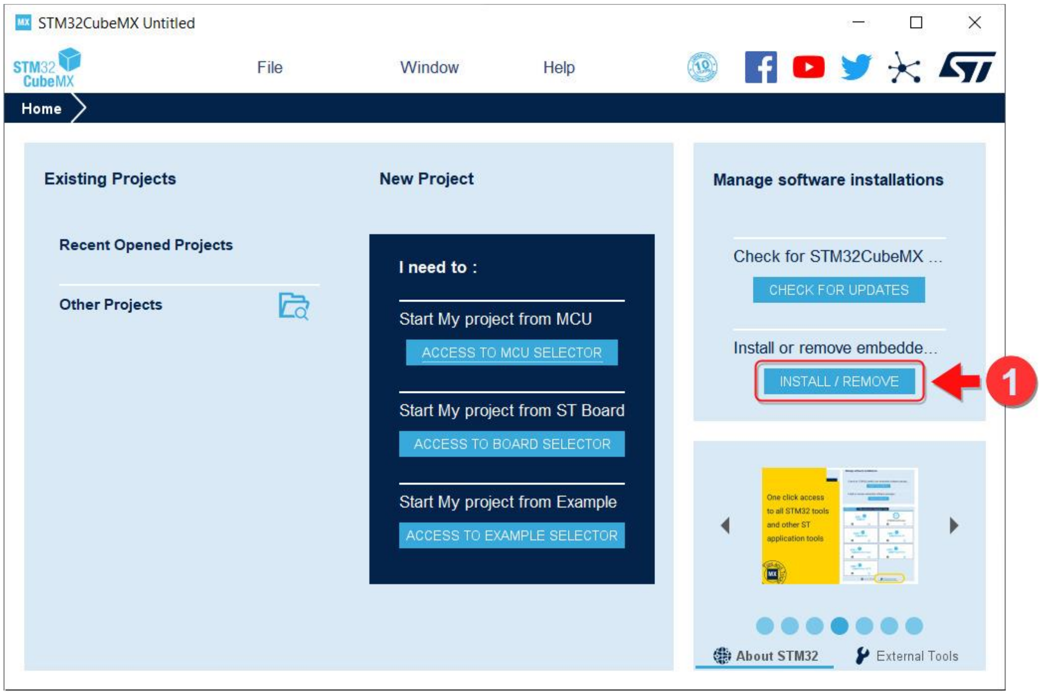
The file name is:

Infineon.Connectivity-STM32.1.0.1.pack

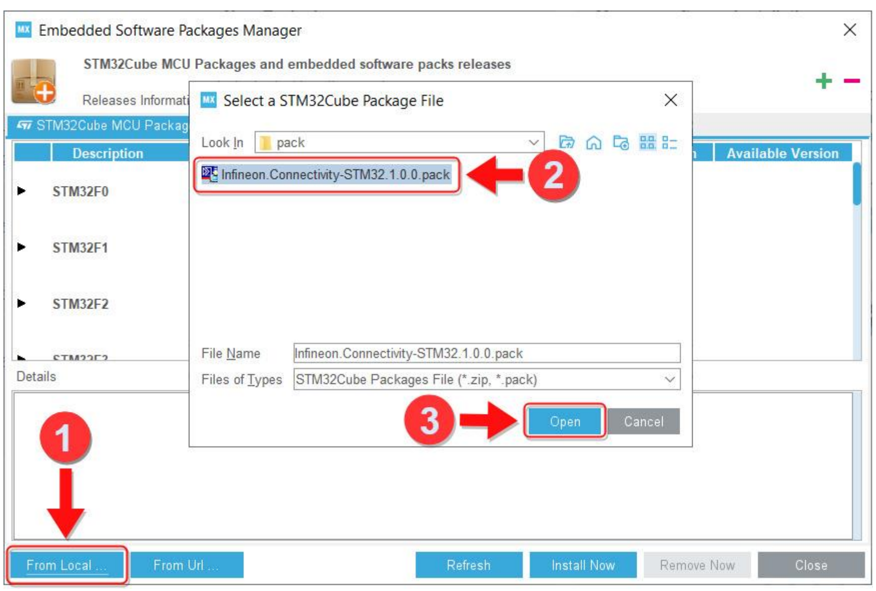
Installing/Importing the Pack

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**.

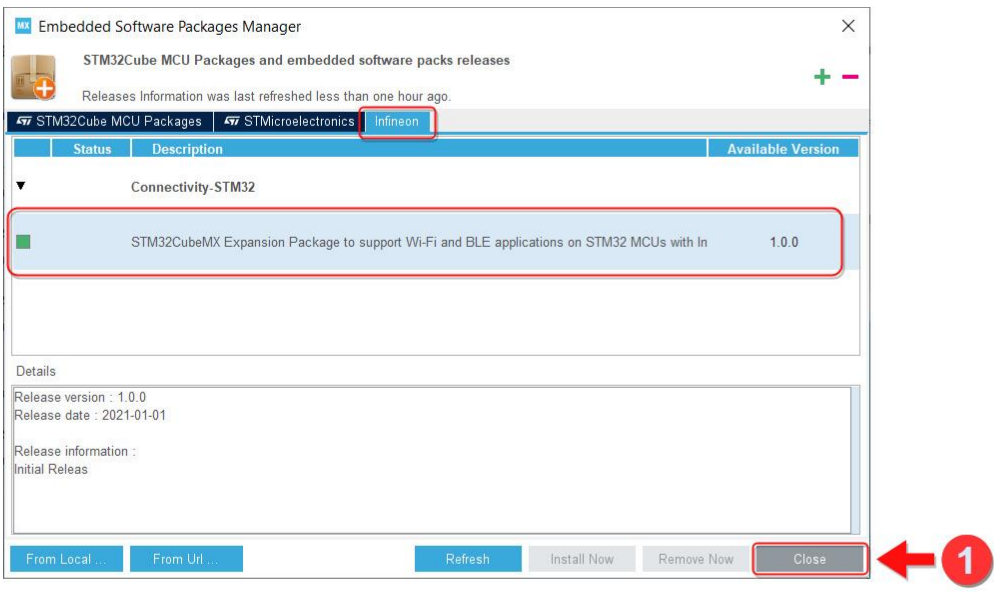


3. Select **From Local…**, navigate to the downloaded pack file, and select **Open**.



4. Accept the license agreement and select Finish.

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



Hardware Setup

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:

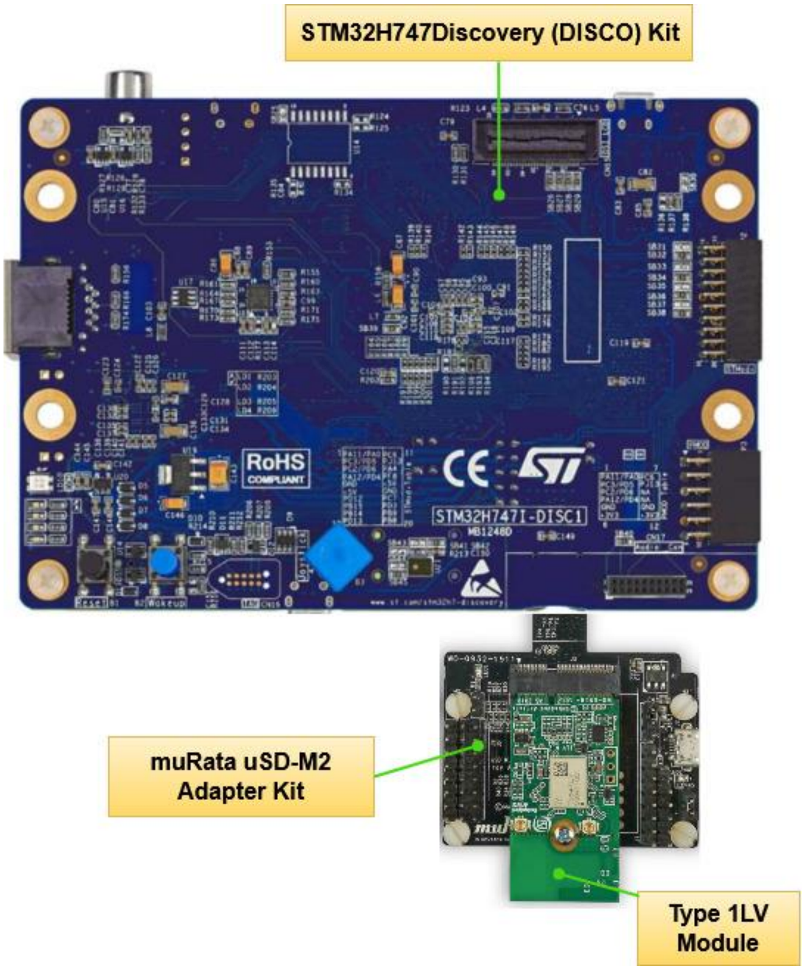
- STM32H747 Discovery (DISCO) Kit: The STM32H747I-DISCO Discovery kit is a complete demonstration and development platform for STMicroelectronics STM32H747XIH6 microcontroller, designed to simplify user application development.
- muRata uSD-M2 Adapter Kit: muRata's uSD-M.2 Adapter Kit with Embedded Artists' 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 Cypress Semiconductor'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.



Set Up Type 1LV Module

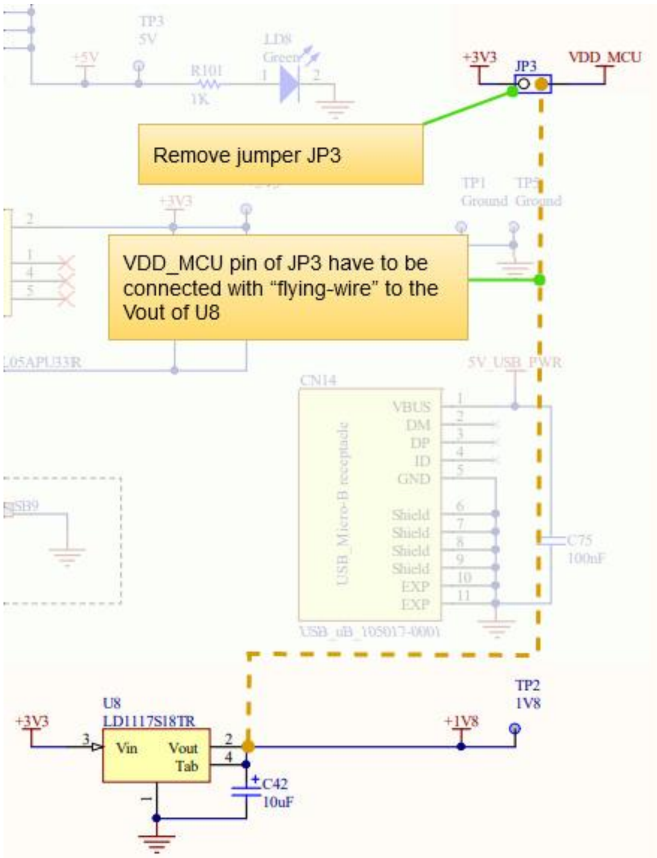
Model	Embedded Artists 1LV M.2 Module	
Features	■ 802.11 a/b/g/n/ac-friendly™ and Bluetooth/LE 5.0 ■ SDIO 3.0 interface, SDR40@80MHz ■ Chipset: Infineon CYW43012	
Datasheet	1LV M.2	

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:

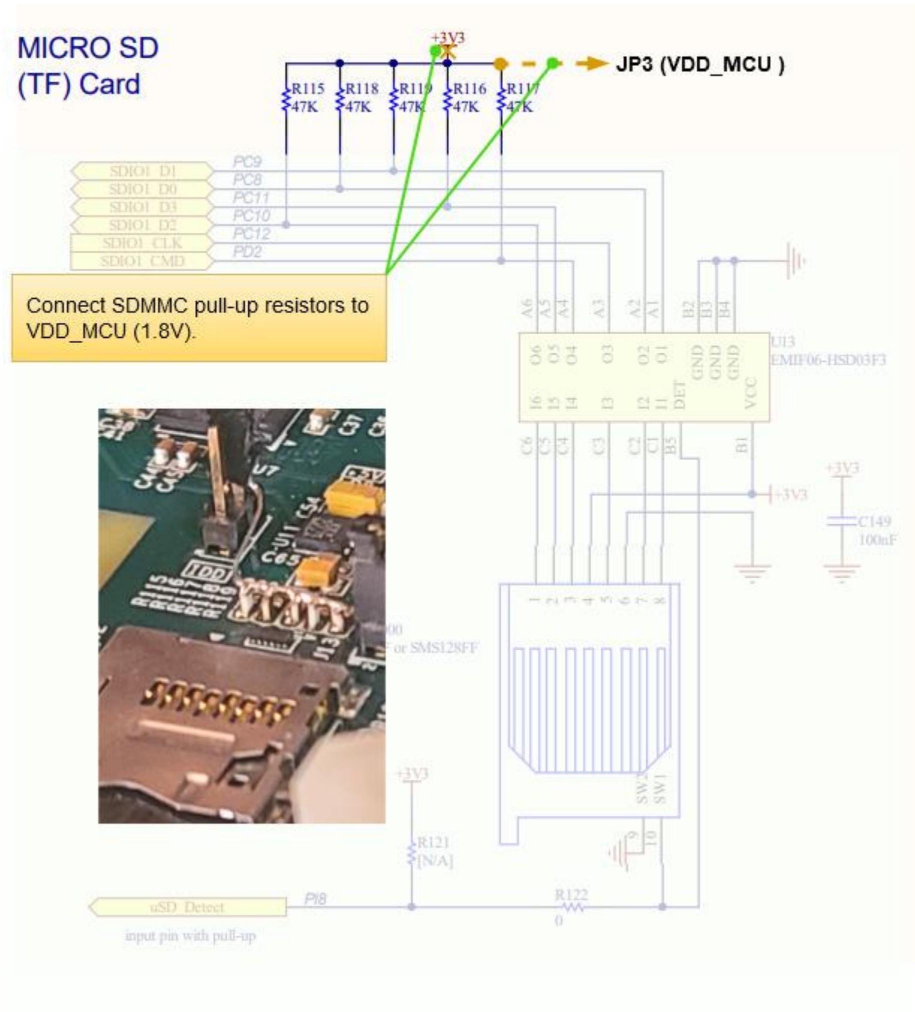
1. Modify STM32H747 Disco Kit to operate on 1.8 V.

   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).



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

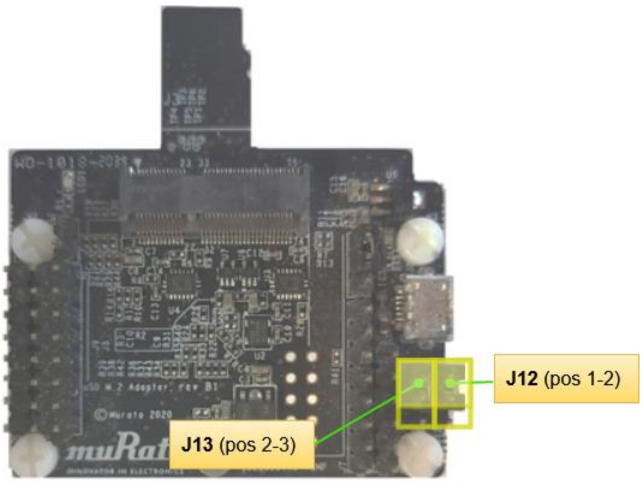
   SDMMC pull-up resistors R115-R119 must be unsoldered from the 3.3 V point and soldered vertically. The tops of these resistors have to be soldered to “flying-wire” and connected to JP3 at the side of VDD_MCU.



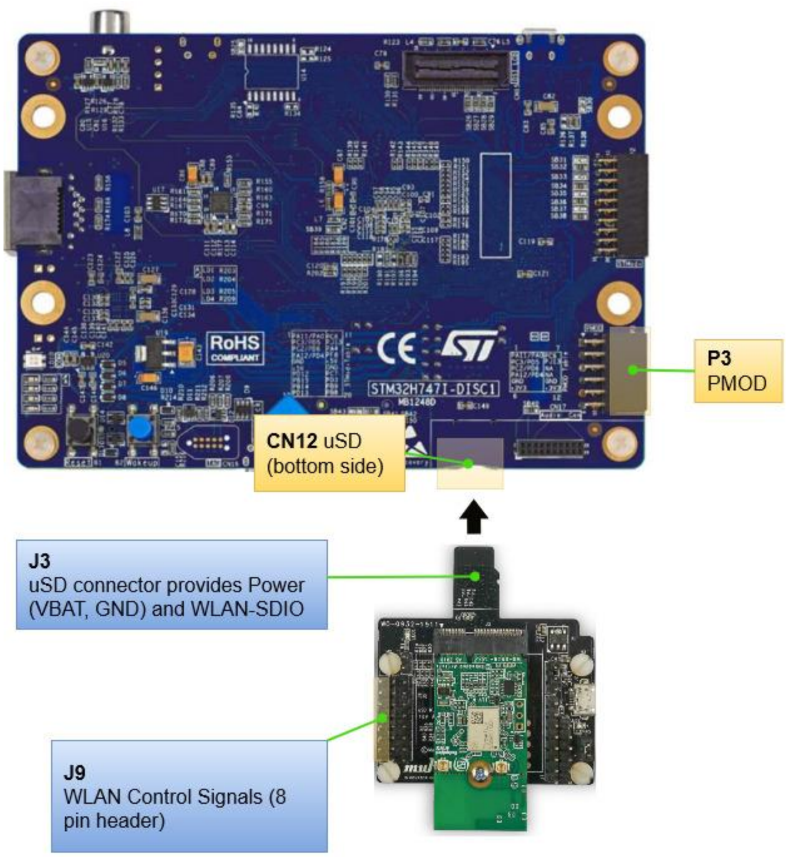
3. Modify muRata uSD-M2 Adapter to operate at 1.8V.

   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)



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				

Connection	Operation	STM32H747 Disco Kit		muRata uSD-M2 Adapter	Note
		Connector	STM32 GPIO		
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)
SDIO	Wi-Fi	CN12	PC8, PC9, PC10, PC11, PC12, PD2	J3	uSD connector pins: provides Power (VBAT, GND) and WLAN-SDIO (DATA0, DATA1, DATA2, DATA3, Clock and Command)

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

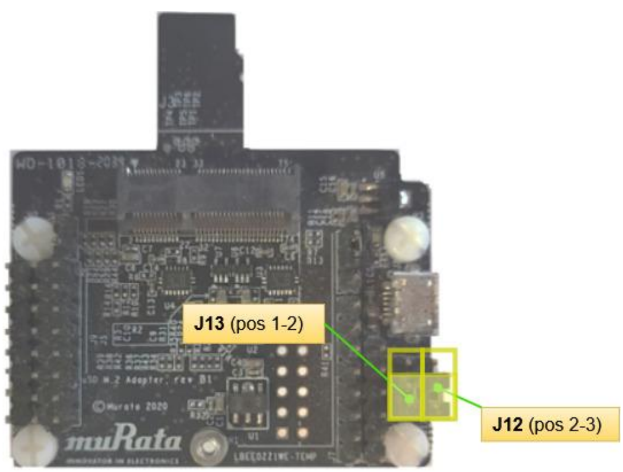
Datasheet [1DX M.2]

Board preparations

This module does not require the host to provide 1.8 V on the SDIO signals. It can operate on 3.3V/1.8V. This makes board preparation simpler.

Ensure that the muRata uSD-M2 Adapter is set to operate at 3.3V, with this jumper configuration:

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



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.

STM32 Wi-Fi Scan Example Project

We provide the following example project to get started using the pack. 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 3 seconds after the previous scan completes.

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

Hardware

Refer to section the STM32 hardware configuration descriptions as appropriate:

Using STM32H747 DISCO Kit

Other Software

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

Project Components

The following are the only components used in this project:

- wifi-host-driver (WHD)
- wifi-connection-manager (WCM)
- abstraction-rtos (configured for the FreeRTOS kernel)
- connectivity-utilities
- core-lib
- pal (minimum interface to ST HAL to enable connectivity)
- whd-bsp-integration
- wifi-mw-core

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\<PACK_VERSION>\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.

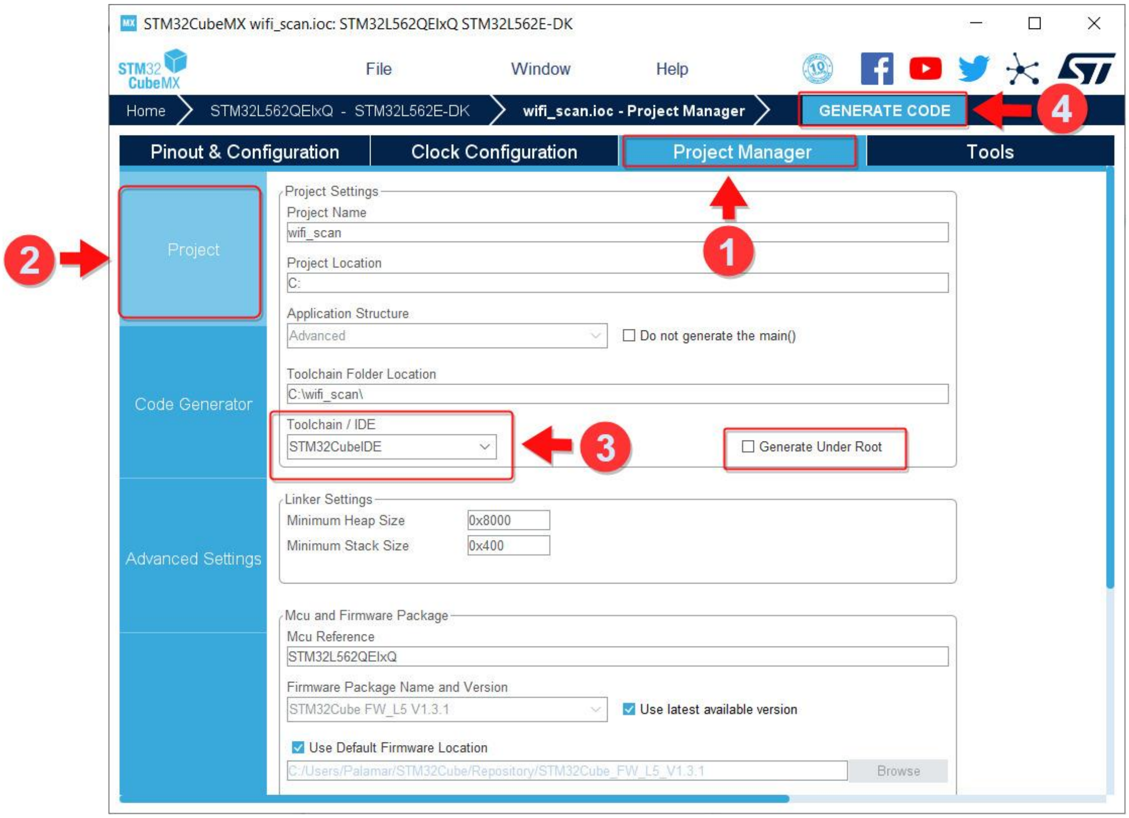
   C:\Users\<USER>\STM32Cube\Example\wifi_scan\wifi_scan.ioc

3. Click **OK** to accept.

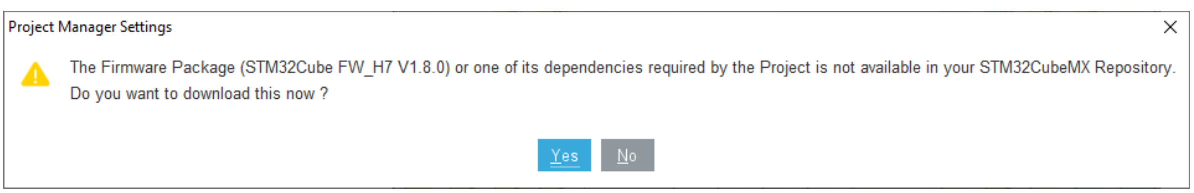
Generate Code

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**:

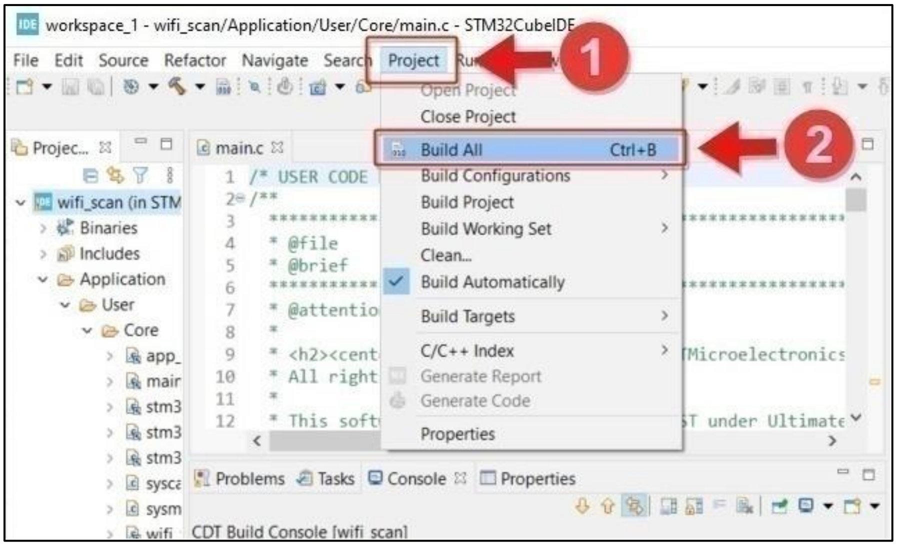


4. After the code is generated, you will see this dialog. Select **Open Project**.

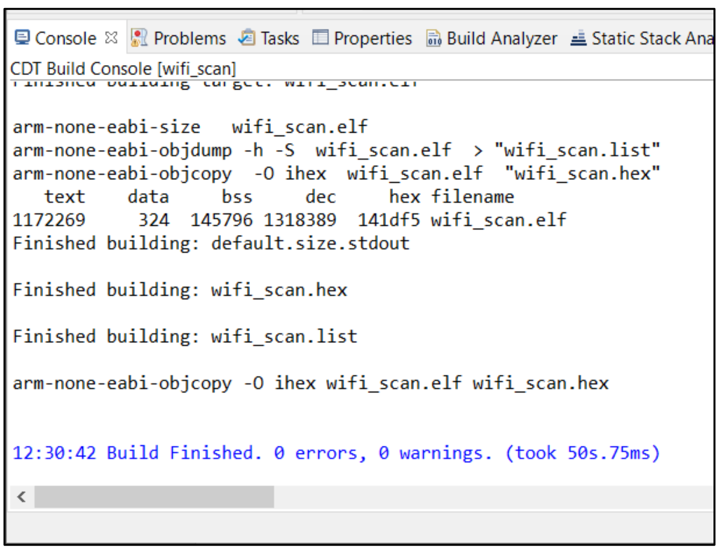
Build the Project

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

STM32CubeIDE:

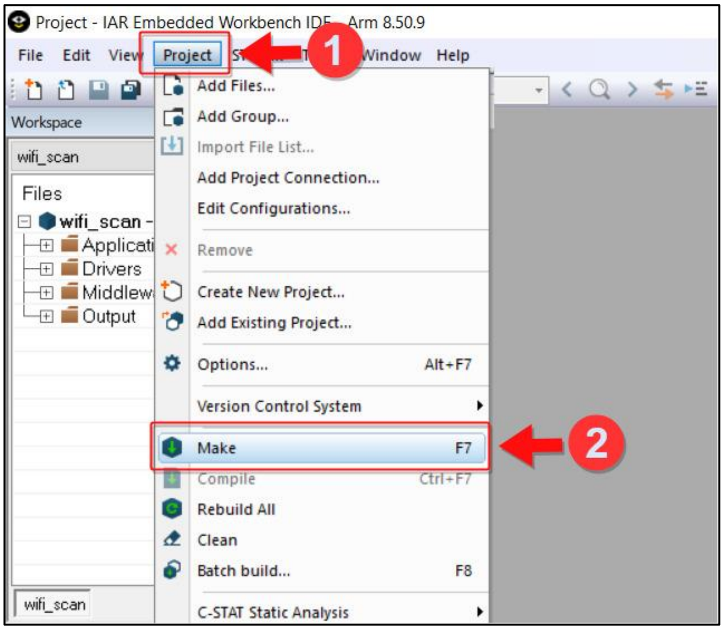


Example output from a successful build:

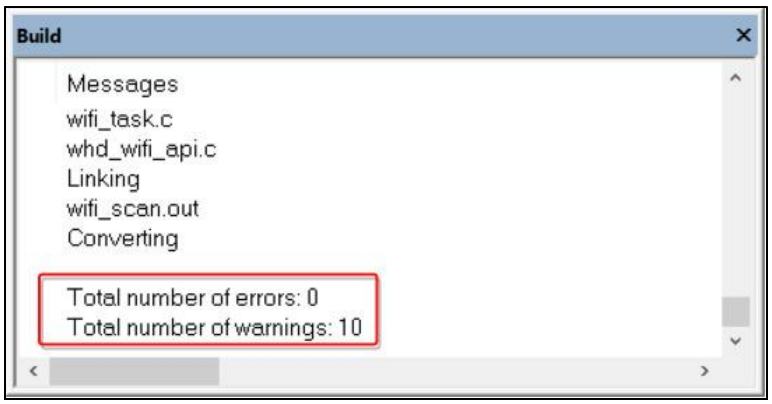


```
Console  Problems  Tasks  Properties  Build Analyzer  Static Stack Ana
CDT Build Console [wifi_scan]

arm-none-eabi-size   wifi_scan.elf
arm-none-eabi-objdump -h -S  wifi_scan.elf  > "wifi_scan.list"
arm-none-eabi-objcopy  -O ihex  wifi_scan.elf  "wifi_scan.hex"
   text    data     bss     dec     hex filename
1172269     324  145796 1318389  141df5 wifi_scan.elf
Finished building: default.size.stdout

Finished building: wifi_scan.hex

Finished building: wifi_scan.list

arm-none-eabi-objcopy -O ihex wifi_scan.elf wifi_scan.hex


12:30:42 Build Finished. 0 errors, 0 warnings. (took 50s.75ms)
```

IAR EWARM:



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



Project Hardware Setup

Refer to section Hardware Setup.

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.

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

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
```

Special Options and Setup

STM32H7xx – Using Serial Flash

There may be a need for extra internal Flash space when running applications on STM32H7xx. The 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 which are needed to enable placing Wi-Fi stack firmware on external memory:

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

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

- 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 = .;
} > QSPI
```

- Preprocessor macro name added:

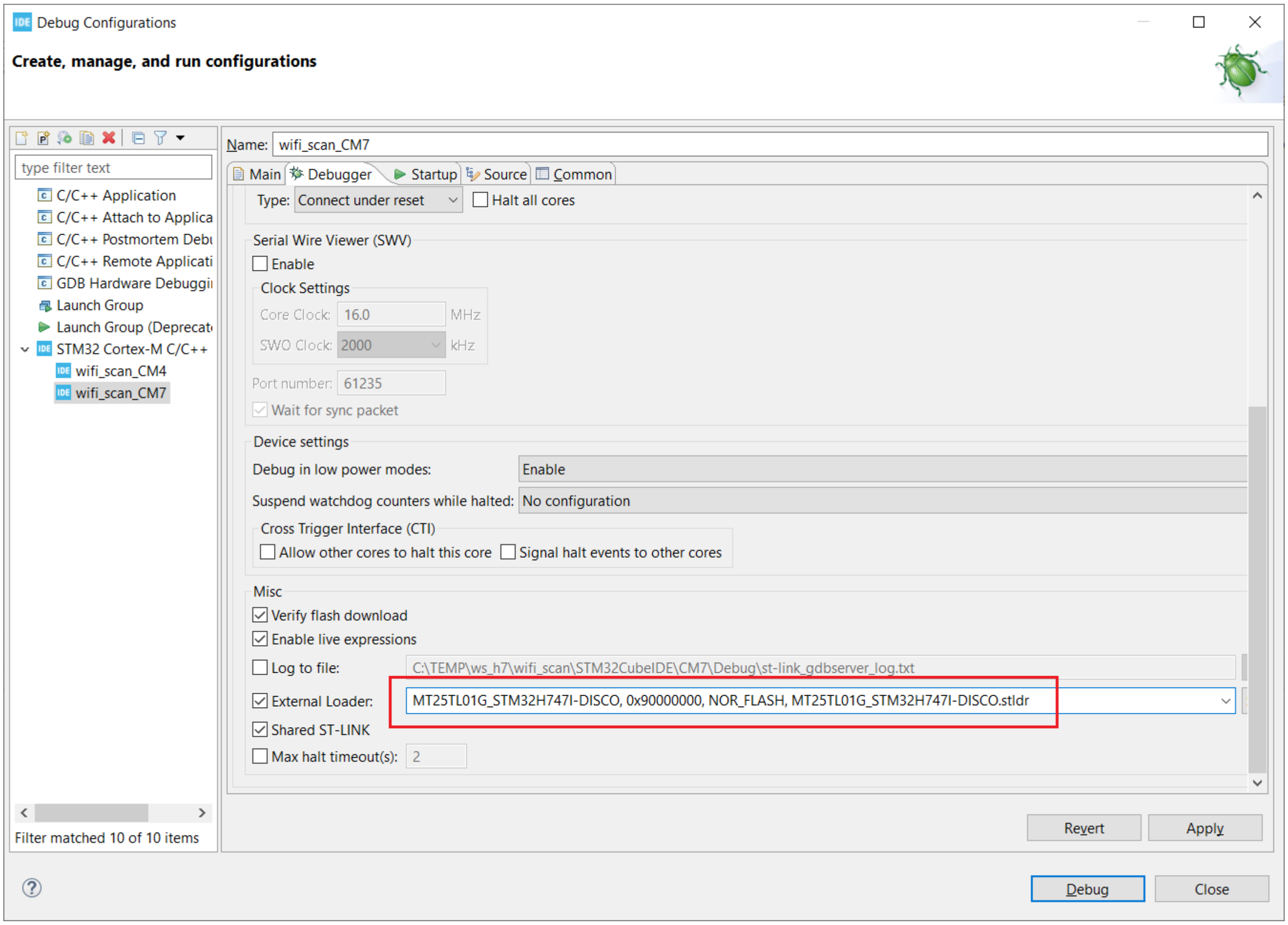
```
CY_STORAGE_WIFI_DATA=".whd_fw"

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
```

- 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();
   }
```

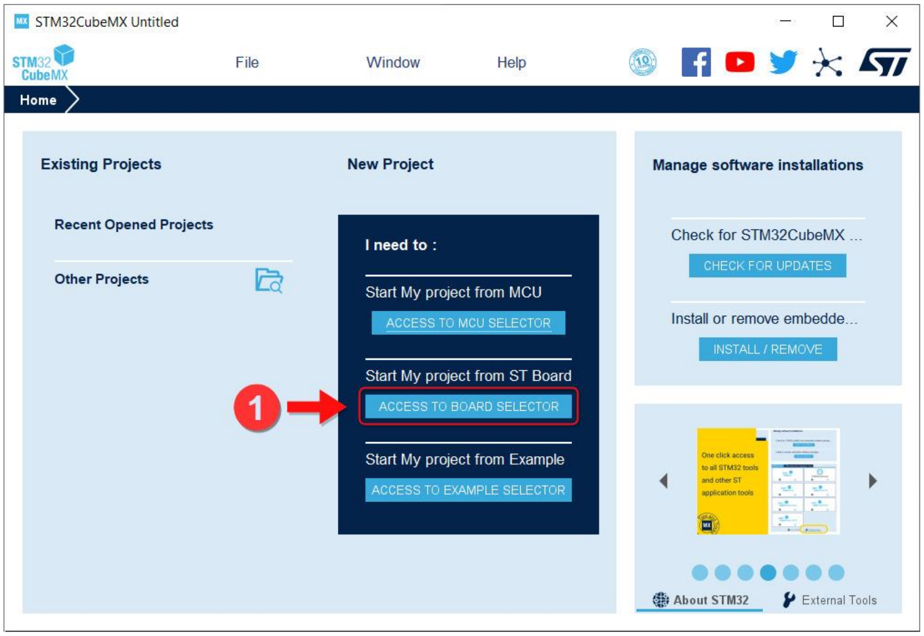
Programming of the Serial Flash should be performed with appropriate Flash Loader selection:



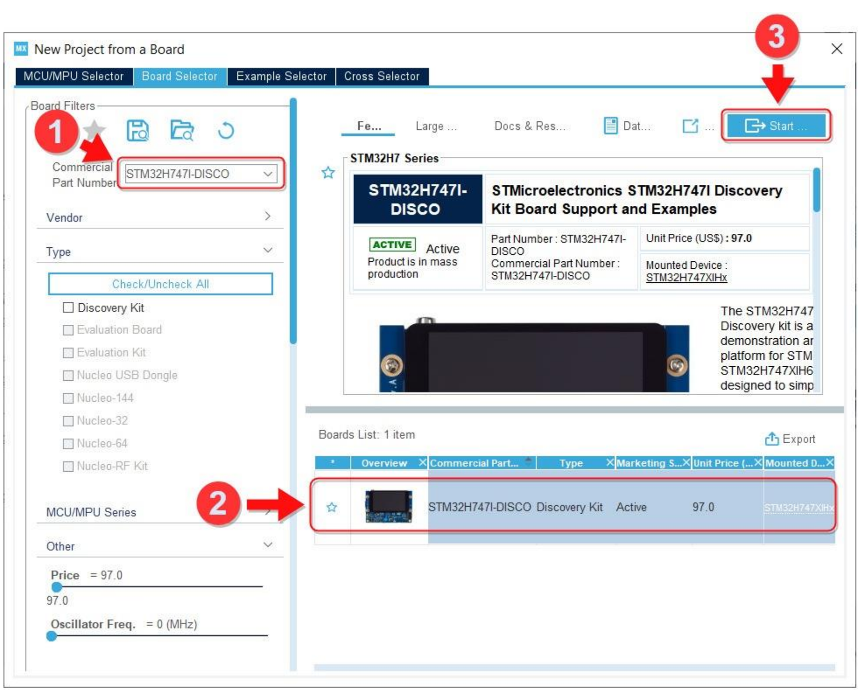
Creating a New Project

The pack should also appear when creating a new project in ST32CubeMX, as illustrated here:

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

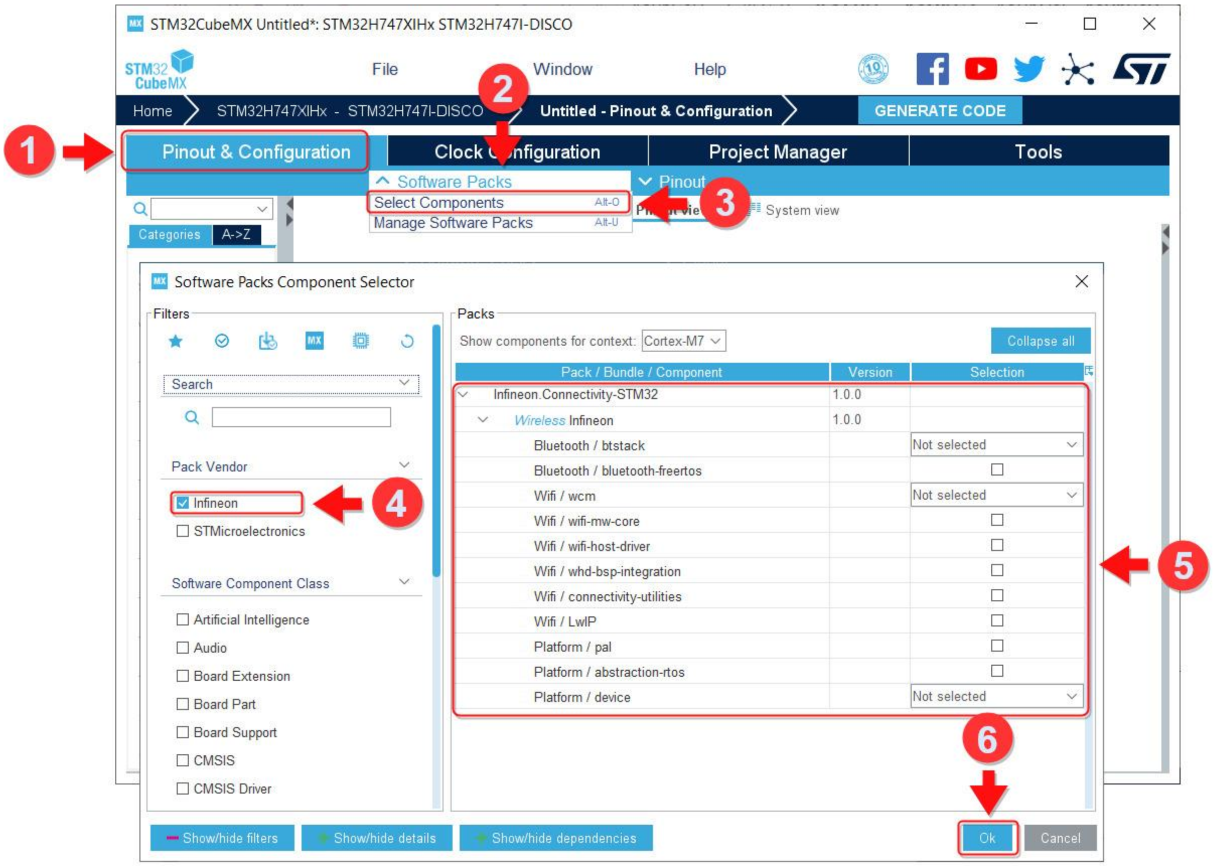


2. Select a board.
   - Enter/select the board number and click on your selected board.
   - Select **Start Project**.



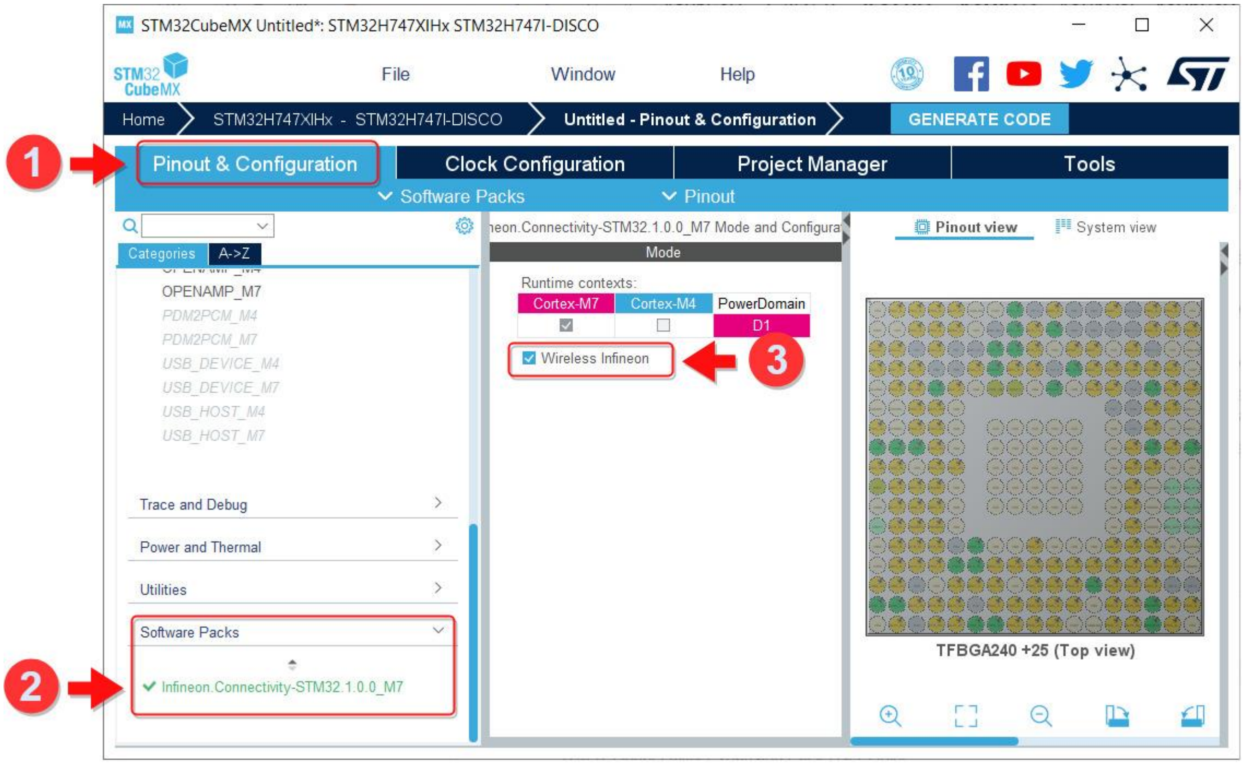
Enable Software components from STM32 Connectivity Expansion Pack

1. Select Software Components.
   - Select the **Pinout & Configuration** tab.
   - Select **Software Packs > Select Components**. This will show a list of the installed packs and their contents.



- Select the components you need for your project – all of the 'Platform' and 'Wifi' components.
  - For the 'Platform / device' component, select the appropriate connectivity device for your system (CYW43012, CYW4343W or CYW43438).
  - For the 'Wifi / wcm' component, select the appropriate variant (LWIP or LWIP/WPS/MBEDTLS).
  - LWIP 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 Cypress platforms that support Wi-Fi connectivity
  - LWIP/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.

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



3. After generating the code, three include files will need to be added to the code base:

- FreeRTOSConfig.h
- lwipopts.h
- cybsp.h

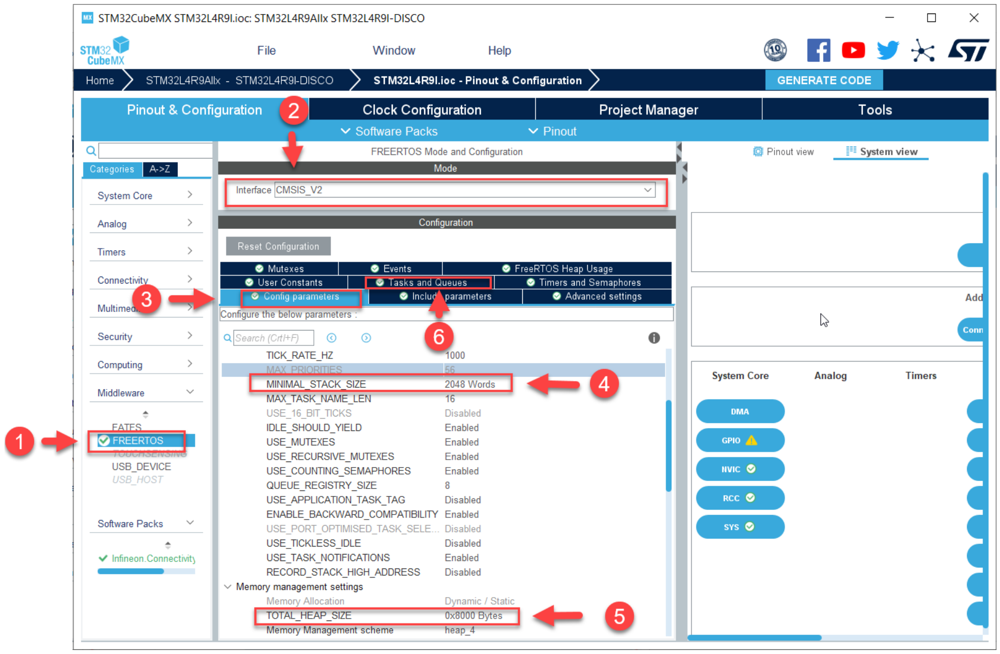
Examples of these files can be found in the appropriate example project in the pack directory. The files can be located in the equivalent directory for the newly created project.

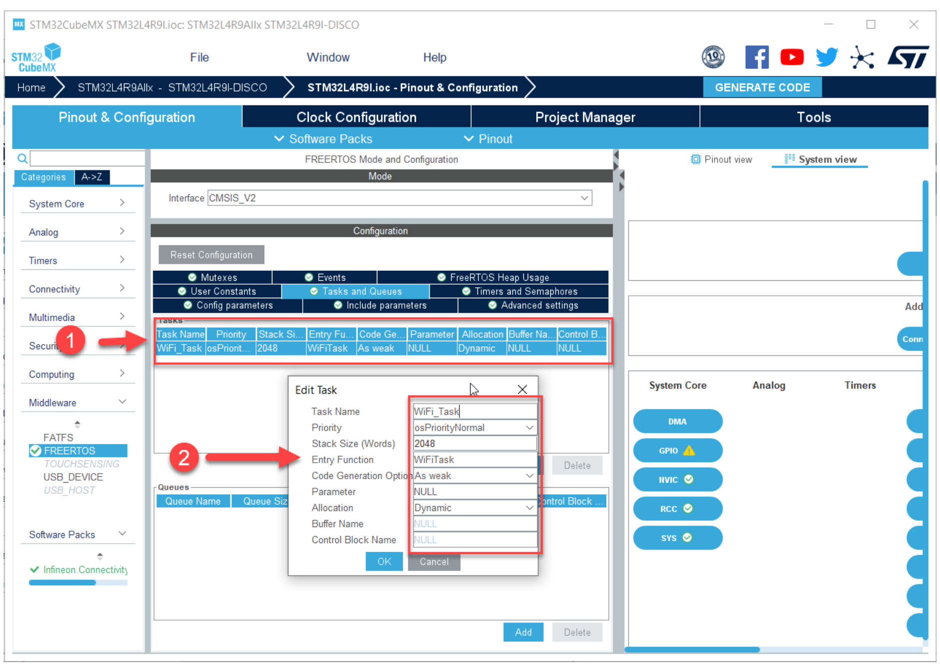
STM32H747I-DISCO:

STM32Cube\Repository\Packs\Infineon\Connectivity-STM32\<PACK_VERSION>\Projects\STM32H747I-DISCO\Applications\wifi_scan\CM7\Core\Inc

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

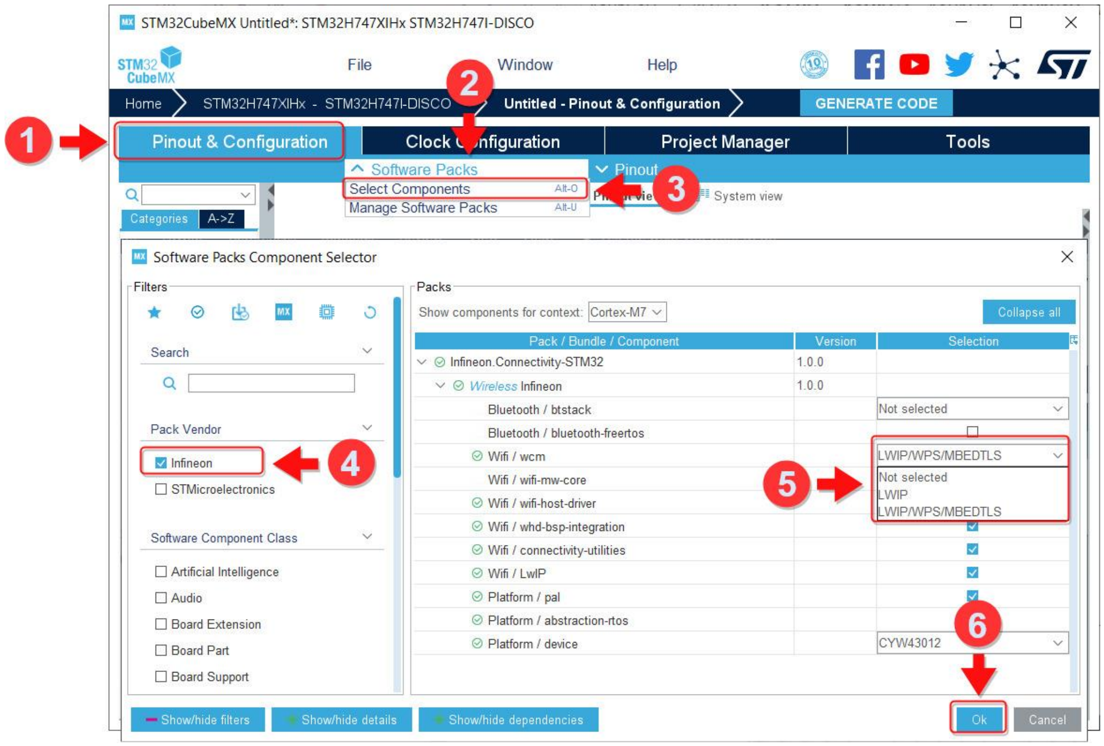




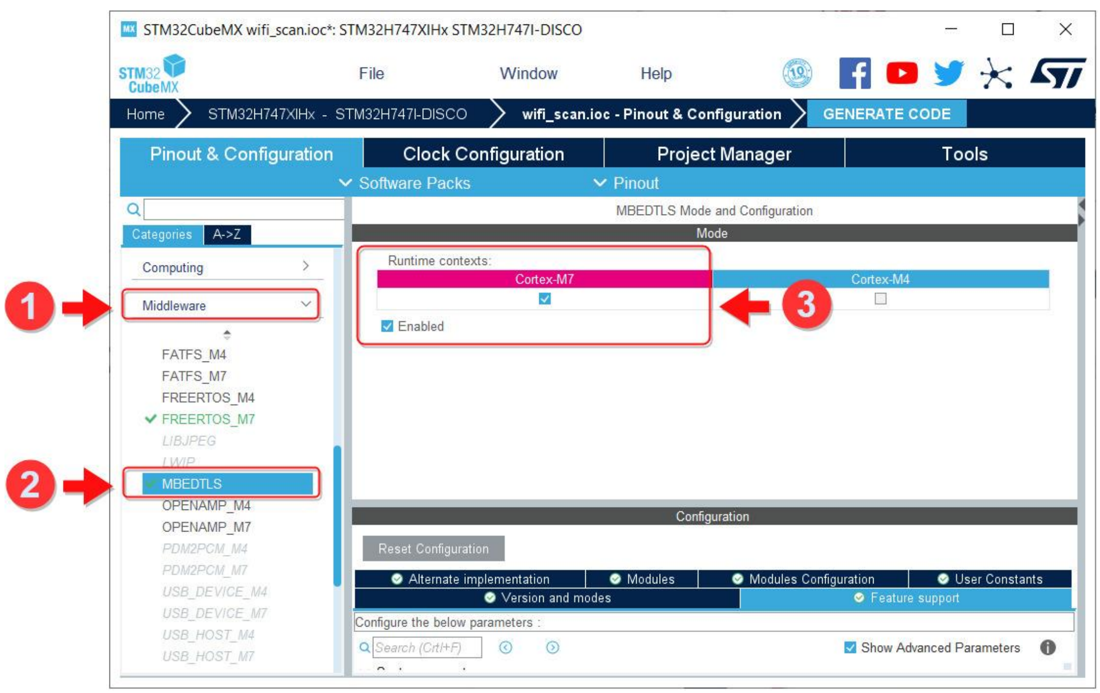
MbedTLS Configuration

The mbedTLS is required by LwIP (Lightweight IP) WCM (WiFi Connection Manager) Pack's components. To enable mbedTLS:

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



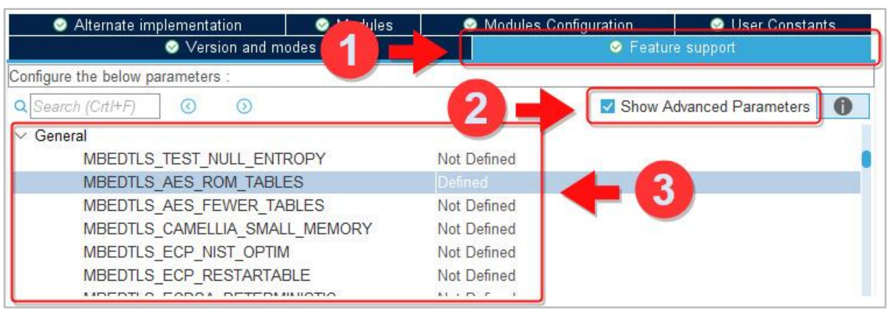
3. Navigate to **Select Components**, select **Middleware** and then select **MBEDTLS** for CM7 and select the **Enabled** check box.



4. Ensure 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:

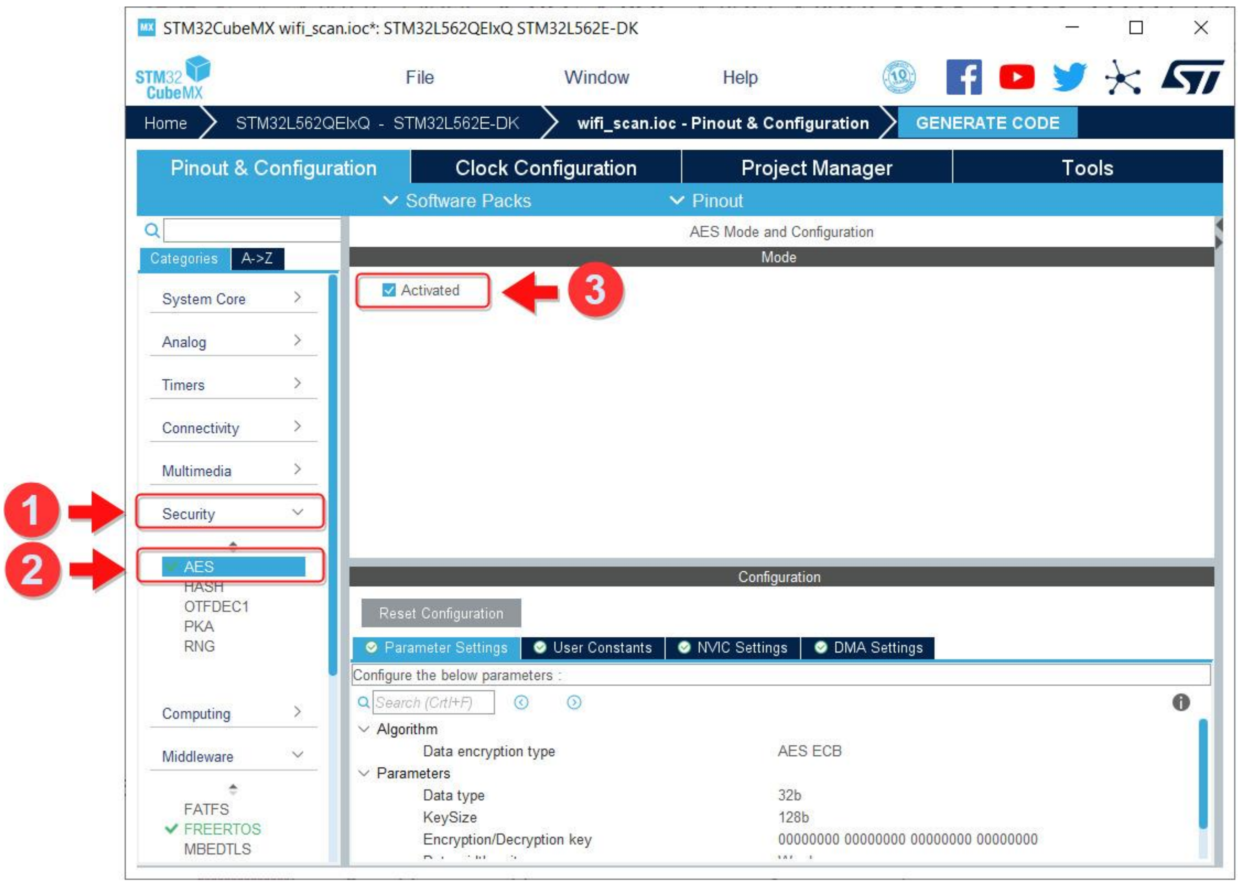
- mbedTLS sources are added to CM7 application
- mbedTLS config is applied to support Infineon's connectivity middleware

Crypto HW Acceleration

STM32 offers HW acceleration for following crypto-related functions:

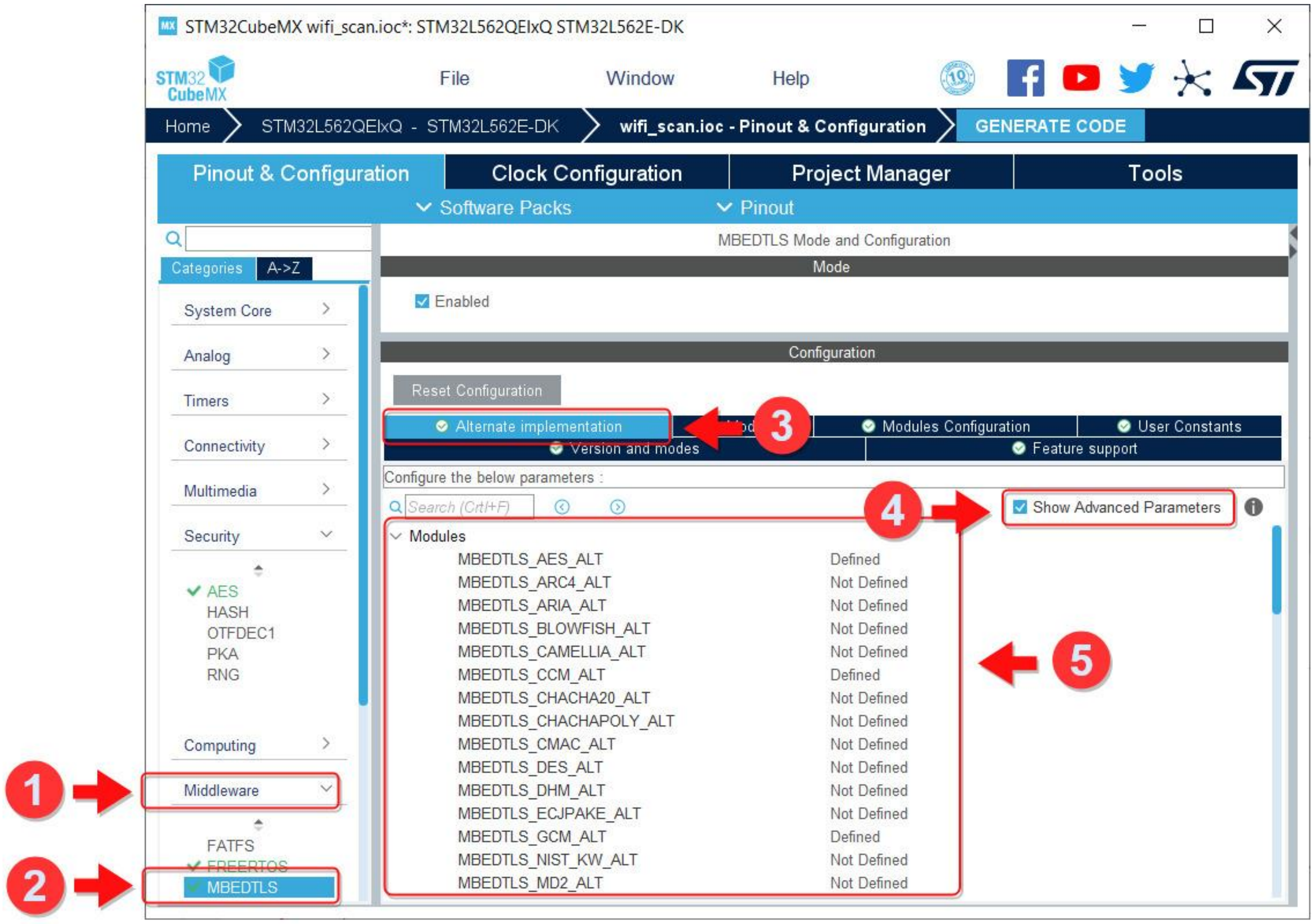
	STM32H7	Notes
RNG	+	
AES		AES-128/256 (ECB, CBC, CTR, GCM GMAC, CCM)
HASH		SHA1, SHA224, SHA256, MD5 HMAC SHA1, HMAC SHA224, HMAC SHA256, HMAC MD5
PKA		Public Key Cryptography
OTFDEC1		On-the-fly decryption of Octo-SPI external memories (AES-128)

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



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

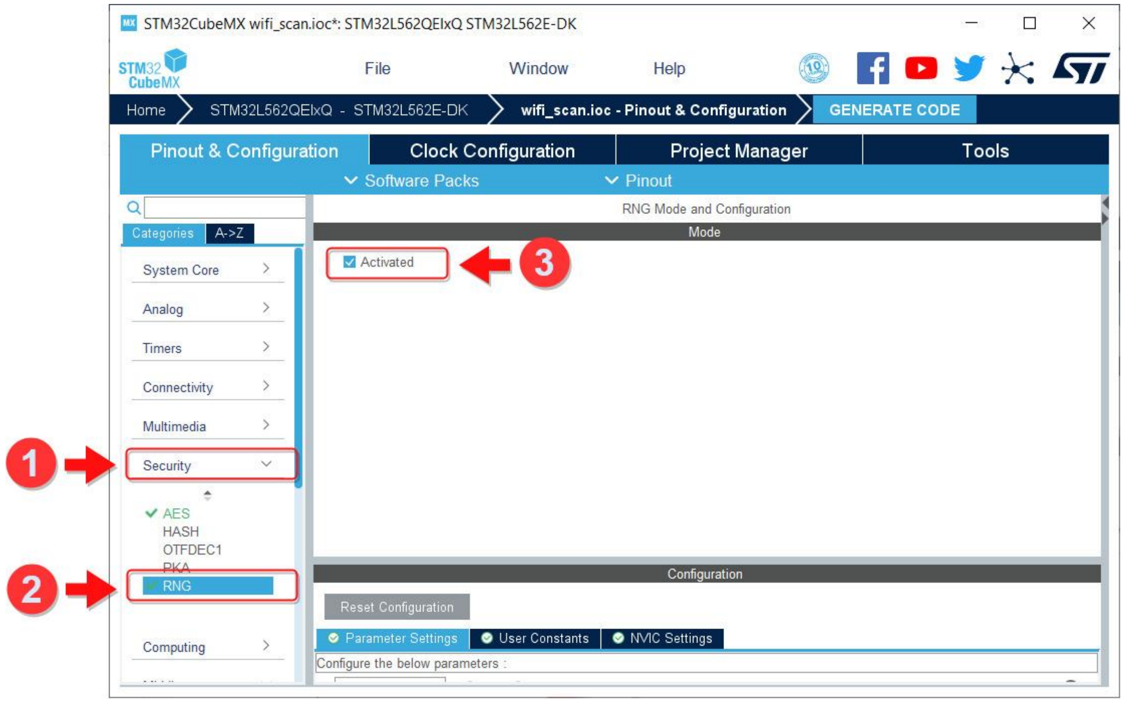


After these steps, source files marked with *_alt suffixes (meaning "alternative", not the original mbedTLS version) will be added into the user's project. They will provide an interface between thembedTLS crypto functions and its HAL HW counterpart.

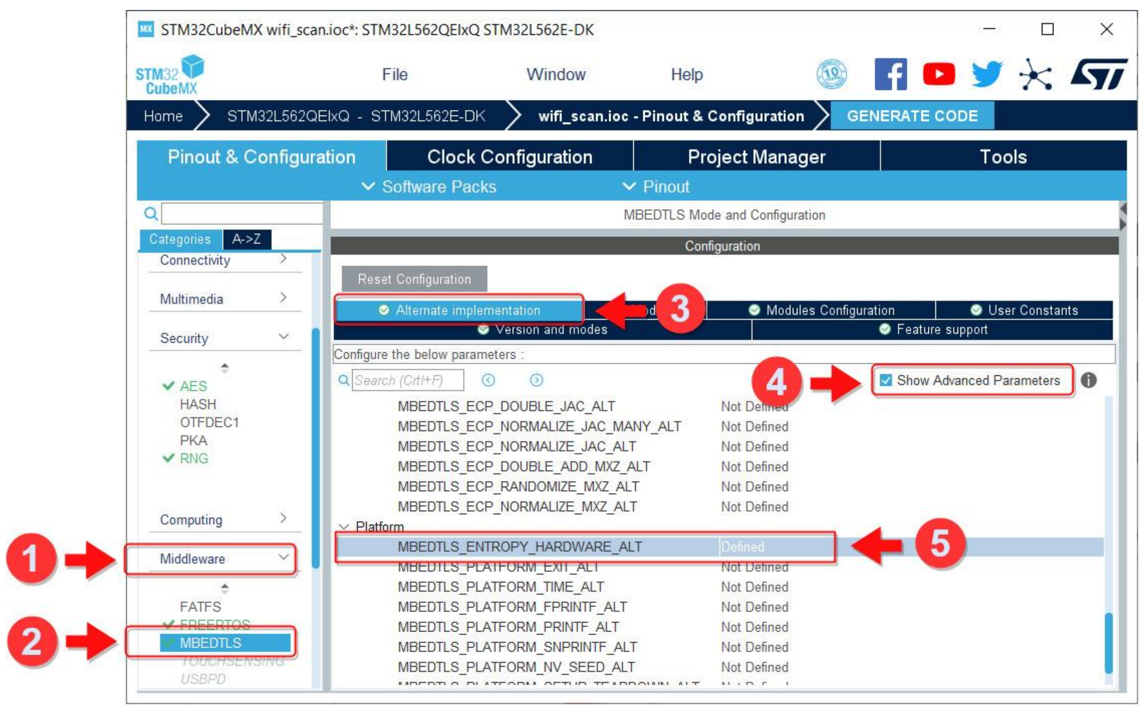
HW Source of Entropy Example

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.

Project Hardware Configuration

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

- SDIO
- Control Pins

Configure resources for WIFI Connectivity

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:

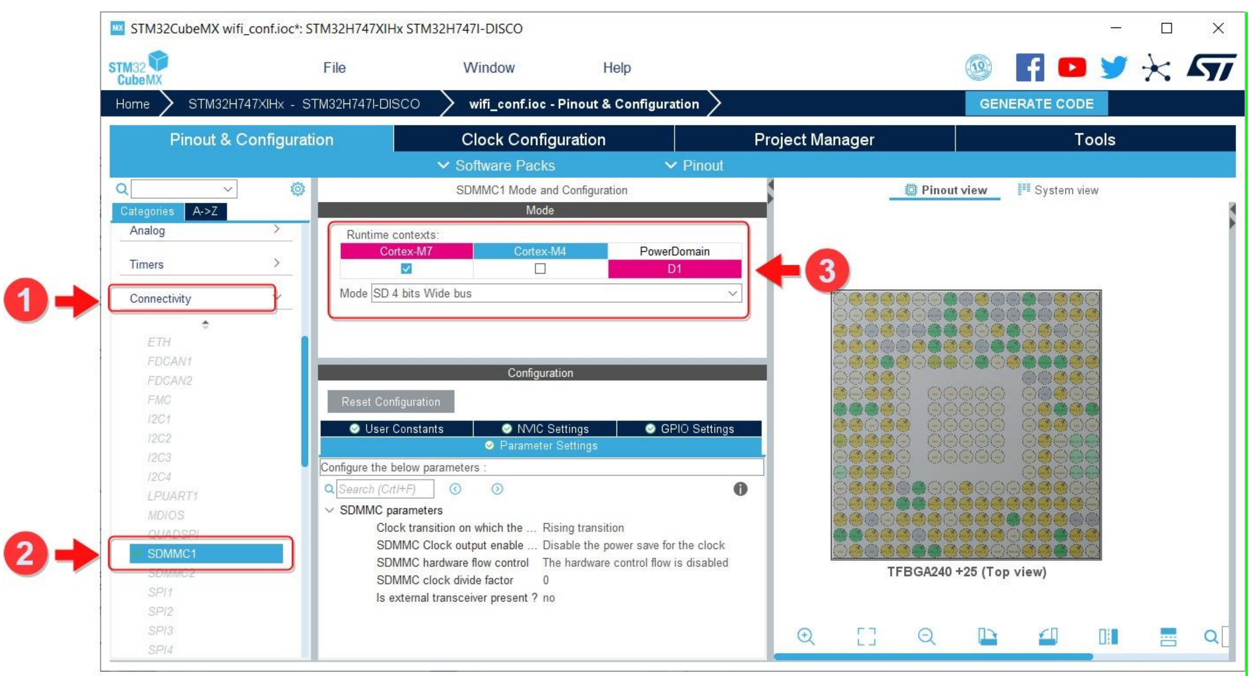
```
SD_HandleTypeDef SDHandle = { .Instance = SDMMC1 };
cy_rslt_t result = stm32_cypal_wifi_sdio_init(&SDHandle);
```

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

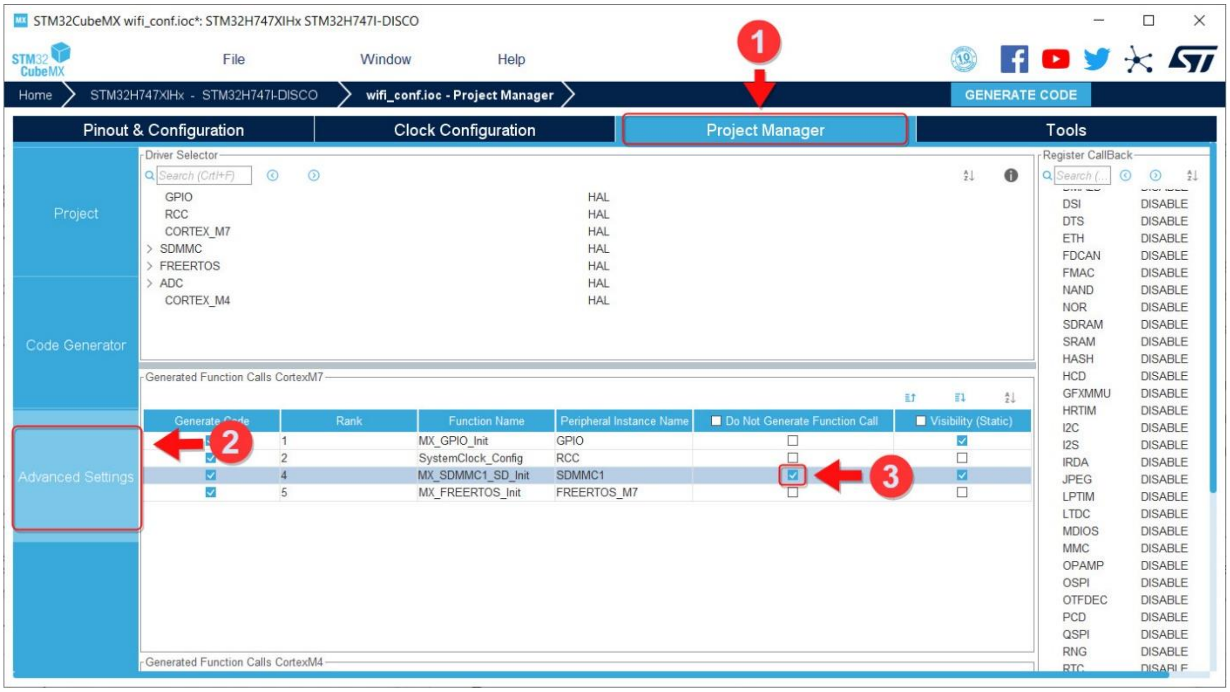
```
void SDMMC1_IRQHandler(void)
{
    stm32_cyhal_sdio_irq_handler();
}
```

Make sure the SDMMC instance selected has its pins routed to the Infineon Connectivity device. The following steps should be done 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).



Control Pins

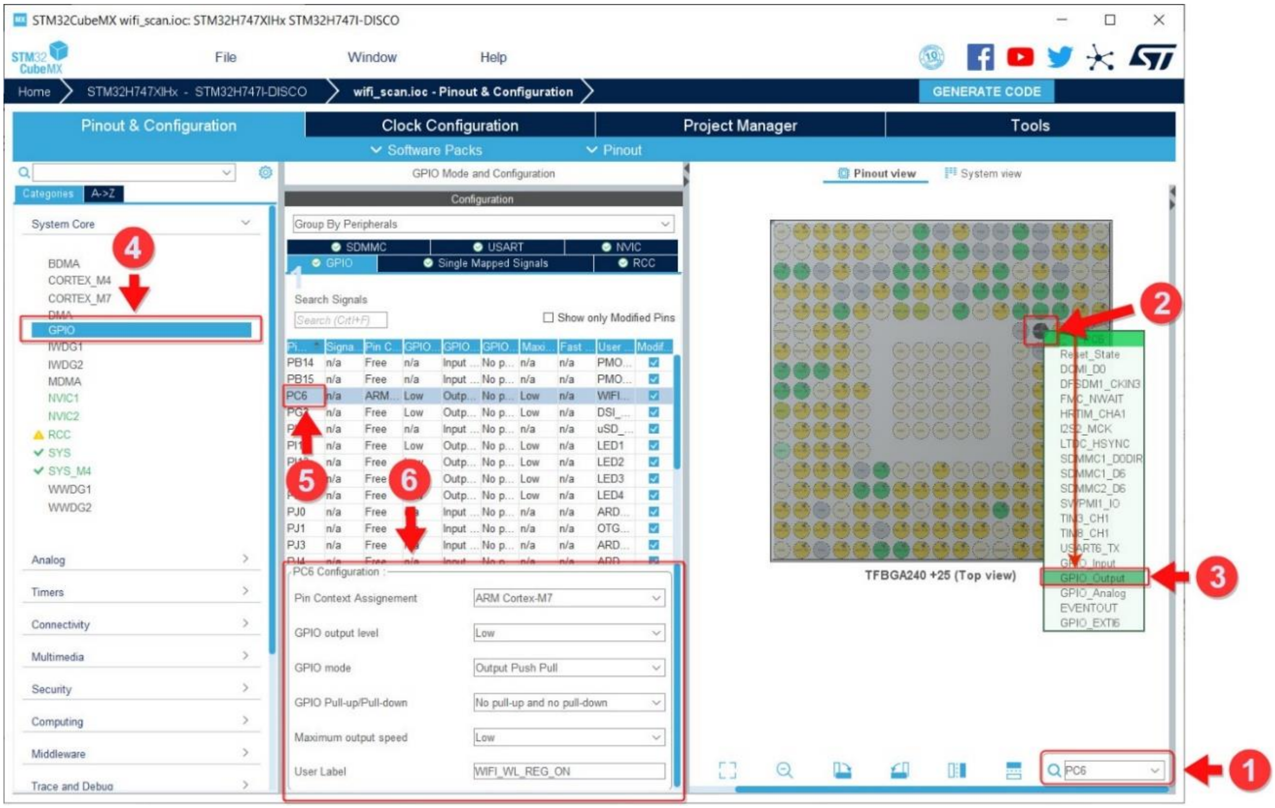
Infineon Connectivity devices require control lines to be connected to host MCU:

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

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	

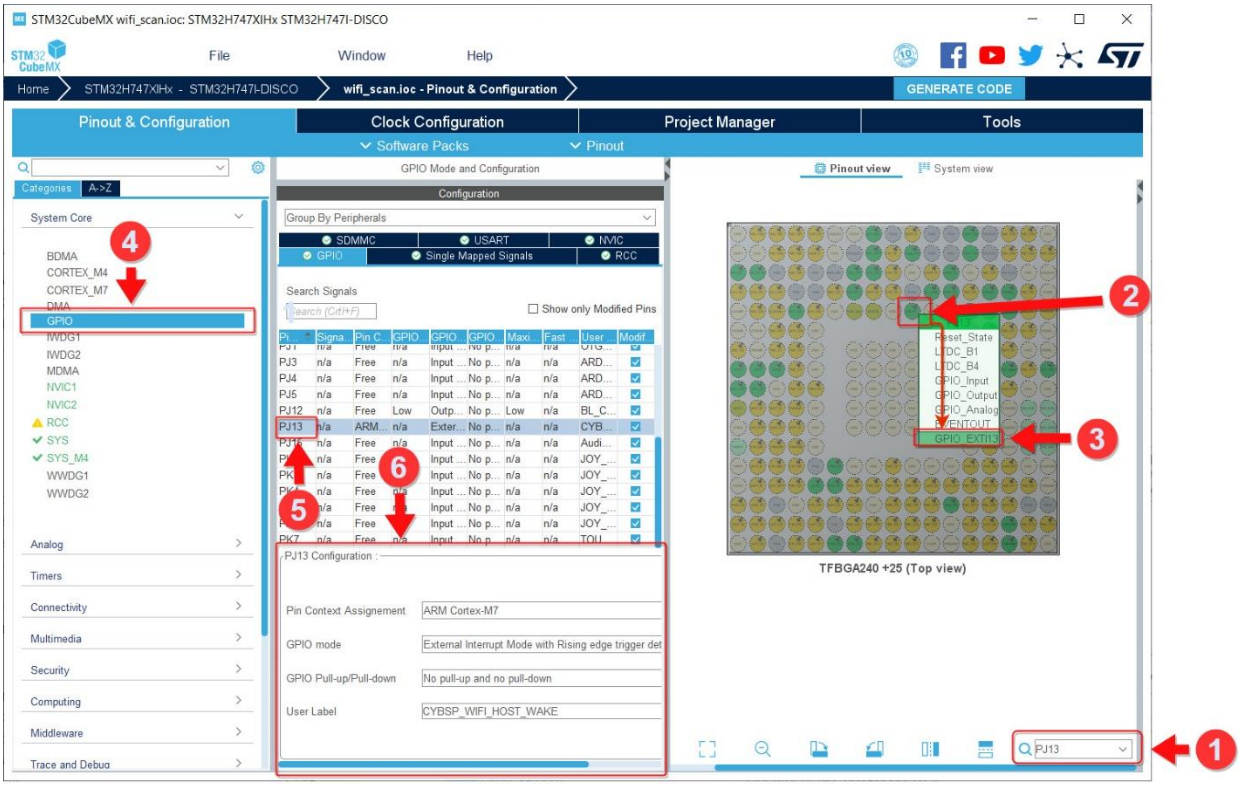


WL_HOST_WAKE

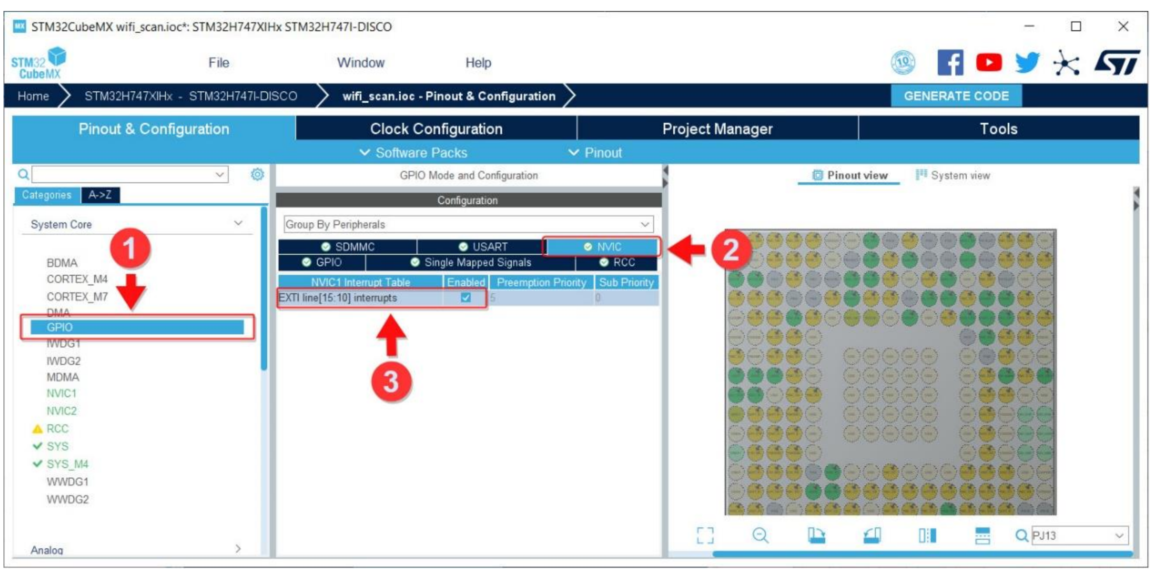
Host MCU Wake signal from WLAN section. WL_HOST_WAKE must be configured in External Interrupt mode / EXTI with following parameters:

GPIO Parameter	Value	Note
Direction	GPIO_EXTI13	
Pin Context Assignment	ARM Cortex-M7	Assign to core, where Connectivity run.
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	

1. Configure in STM32CubeMX:



2. Enable NVIC interrupt for EXTI line:

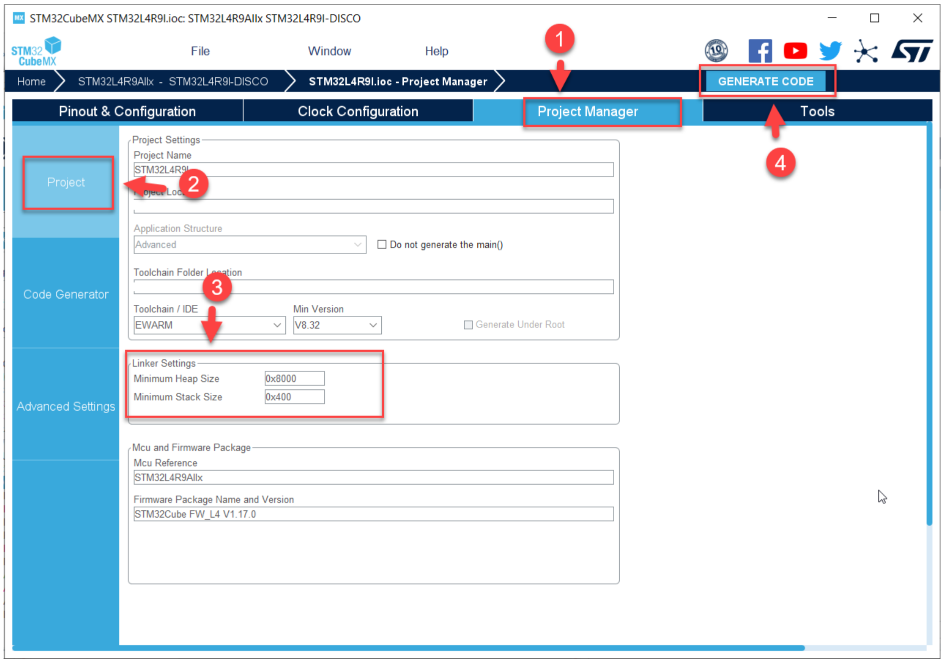


3. EXTI Callback handler must be overwriting in application and call stm32_cyhal_gpio_irq_handler function:

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

Heap and Stack Configuration

Configure Heap and Stack size required for the example app.



Generating the 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\<PACK_VERSION>\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 on Configuring Control pins section

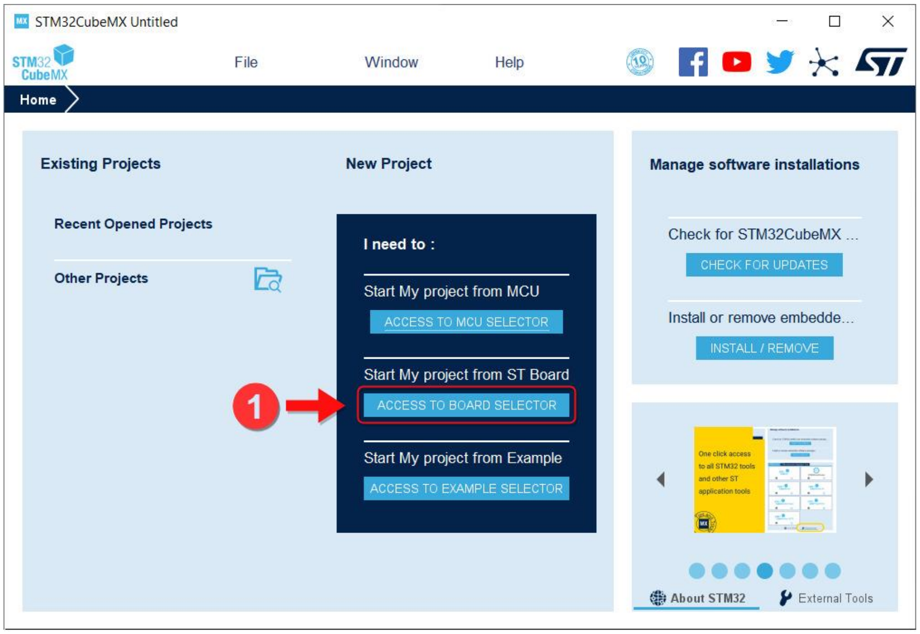
```
/** These names are explicitly referenced in the support libraries */
#define  CYBSP_WIFI_WL_REG_ON           ***
#define  CYBSP_WIFI_HOST_WAKE           ***
```

Example Project for Non-H7 MCU Boards

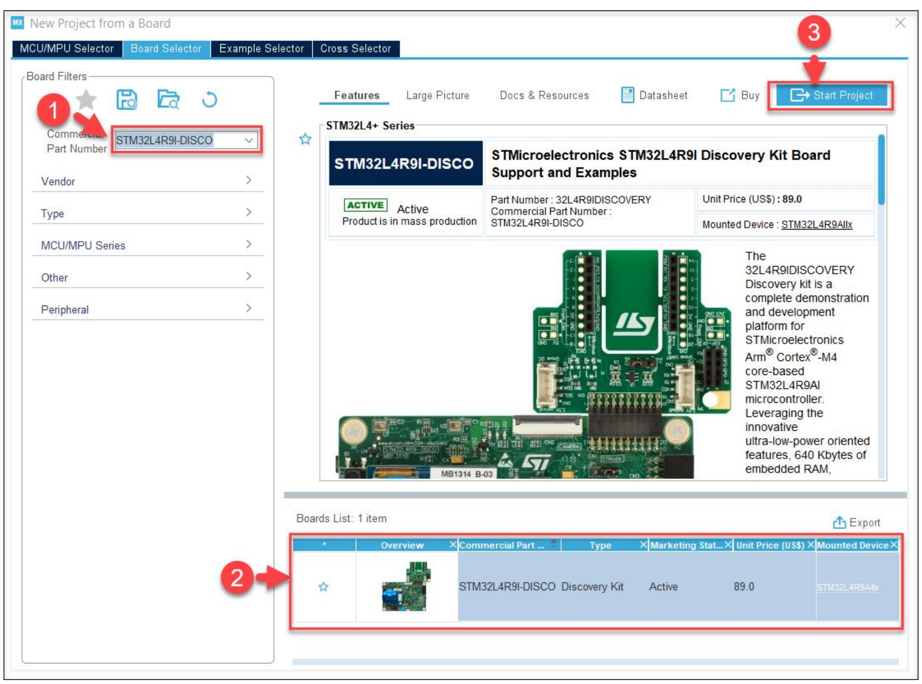
This Section explains how to create new example project for any non-H7 MCU boards using the expansion pack.

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.

FreeRTOS Configuration

Follow same steps as mentioned in FreeRTOS Configuration.

Other Configurations

- Configure SDMMC (refer to SDIO).
- Configure Control Pins (refer to Control Pins).
- Configure Heap and Stack size (refer to Heap and Stack Configuration).

Changes required in PAL library

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

- 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"
```

- 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
```

- stm32_cyhal_gpio.c

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

Changes required in main.c

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

- 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);
```

- 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();
}
```

- 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);
}
```

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

OctoSPI Configuration

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

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

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

- 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
```

- Add Preprocessor macro name:

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