Getting started with the STMicroelectronics BlueNRG-MS software package for STM32CubeMX

Introduction

This document provides the guidelines to configure and use the BlueNRG-MS software package for STM32CubeMX (minimum required version V4.26.0). The document contains a description of the provided sample applications, a description of the steps required to configure a generic project using the BLE middleware, as well as a description of the steps to configure and use the sample application provided in the package.

Information and documentation related to the ST BlueNRG-MS network processor, the X-NUCLEO-IDB05A1 expansion board and the ST expansion software for Bluetooth Low Energy are available on www.st.com.

Contents

List of figures

1 Acronyms and abbreviations

Table 1: list of acronyms

Acronym	Description
BLE	Bluetooth Low Energy
HAL	Hardware Abstraction Layer
HID	Human Interface Device
IOT	Internet Of Things
IP	Internet Protocol
LAN	Local Area Network
NVIC	Nested Vectored Interrupt Controller
PCB	Printed Circuit Board
RTC	Real Time Clock
RTOS	Real Time Operating System
SPI	Serial Peripheral Interface
UID	Unique Identifier
URL	Uniform Resource Locator
U(S)ART	Universal (Synchronous) Asynchronous Receiver Transmitter
USB	Universal Serial BUS
TCP	Transmission Control Protocol

2 What is STM32Cube?

STM32Cube™ represents an original initiative by STMicroelectronics to ease developers' life by reducing development effort, time and cost. STM32Cube covers the STM32 portfolio. Version 1.x of STM32Cube includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as the STM32CubeF4 for STM32F4 series).
    - STM32Cube HAL, an STM32 abstraction layer embedded software, ensuring maximized portability across the STM32 portfolio;
    - a consistent set of middleware components, such as RTOS, USB, TCP/IP, graphics;
    - all embedded software utilities, including a full set of examples.

3 License

The software provided in this package is licensed under Software License Agreement SLA0077.

4 Sample Applications Description

4.1 SensorDemo

This sample application, that could be considered the BlueNRG-MS “Hello Word” application, shows how to use the BlueNRG-MS expansion board to send data from an STM32 Nucleo board to a BLE device such as a smartphone. In this example the stack composed by the STM32 and the BlueNRG-MS expansion boards act as Server-Peripheral, while the smartphone act as Client-Central. On the smartphone it is required to have installed and running the **BlueNRG** app, freely available on both Play Store and iTunes.

After establishing the connection between the STM32 board and the smartphone:

- by pressing the USER button on the board, the cube showed by the app on the smartphone in the MOTION tab will rotate on the three axes (x,y,z);
- in the ENVIRONMENT tab of the app the temperature, pressure and humidity emulated values sent by the STM32 Nucleo to the smartphone are shown;
- in the OTHER tab of the app the RSSI value is shown.

4.2 SampleApp

This sample application shows how to simply use the BLE Stack.

To test this application you need two STM32 Nucleo boards with their respective BlueNRG-MS STM32 expansion boards. One board needs to be configured as Server-Peripheral role, while the other needs to be configured as Client-Central role. Before flashing the boards, please make sure to use the right configuration by enabling/disabling the *#define SERVER_ROLE* in file *app_bluenrg-ms.c*.

Once the two STM32 Nucleo boards have been configured (one as Client and the other as Server) and flashed, the connection between the two boards establishes (when the LED2 on the CLIENT turns off).

By pressing the USER button on one board, the LD2 LED on the other one gets toggled and viceversa.

If you have only one STM32 Nucleo board, you can program it as SERVER and use as CLIENT the BLE IOT app for Android devices.

4.3 Beacon

This example application shows how to use the BlueNRG Bluetooth Low Energy (BLE) expansion board to implement an Eddystone Beacon device.

An Eddystone Beacon is a smart Bluetooth Low Energy device that transmits a small data payload at regular intervals using Bluetooth advertising packets.

Beacons are used to mark important places and objects. Typically, a beacon is visible to a user's device from a range of a few meters, allowing for highly context-sensitive use cases.

Eddystone is an open beacon format from Google that works with Android and iOS.

Two different kinds of devices can be selected through *#define EDDYSTONE_BEACON_TYPE* in file *app_bluenrg-ms.c*:

- EDDYSTONE_UID_BEACON_TYPE: a UID beacon broadcasts a unique ID that provides proximity and general location information.
- EDDYSTONE_URL_BEACON_TYPE: a URL beacon broadcasts a packet containing an URL code usable by compatible applications.

To locate the beacon, it is necessary to have a scanner application running on a BLE-capable smartphone, such as one of the following ones for Android:

- Physical Web
- iBeacon & Eddystone Scanner
- Beacon Radar

An alternative is to use a *Physical Web* compatible browser like Google Chrome (version ≥ 44).

4.4 VirtualCOMPort

VirtualCOMPort is the application to be used for updating the BlueNRG-MS firmware on the X-NUCLEO-IDB05A1 expansion boards.
It must be used along with the flashUpdater java tool enclosed in the generated application folders at:
Middlewares\ST\BlueNRG-MS\Utilities\PC_Software\FlashUpdaterTool
The User can also use this sample application in order to port the BlueNRG-MS VCOM application to his specific BlueNRG-MS PCB (assuming that the customer PCB has a USB or RS232 I/O port available for PC connection).
This application provides an interface compliant with the Bluetooth Low Energy DTM test commands. Anyway, this application is not a reference application to be used for BlueNRG-MS application development and evaluation.

5 Starting a new project

After launching the STM32CubeMX, click the New Project writing in the GUI.

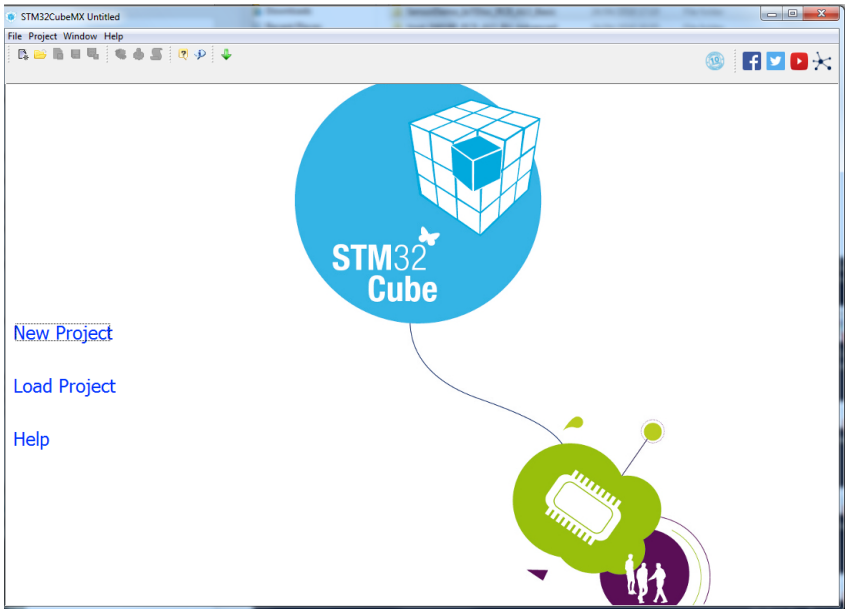

Figure 1 *STM32CubeMX main page*

The **MCU/Board selector** window will pop up. From this window, the STM32 MCU or platform can be selected.

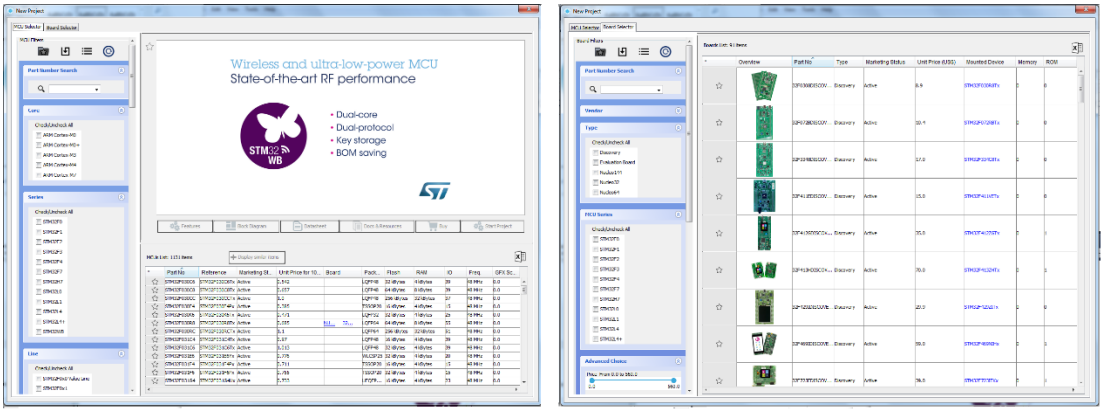

***Figure 2** STM32CubeMX **MCU/Board Selector** windows*

After selecting the MCU or the Board, the selected STM32 pinout will appear. From this view the user can set up the project, by adding one or more Additional Software and peripherals and configuring the clock.

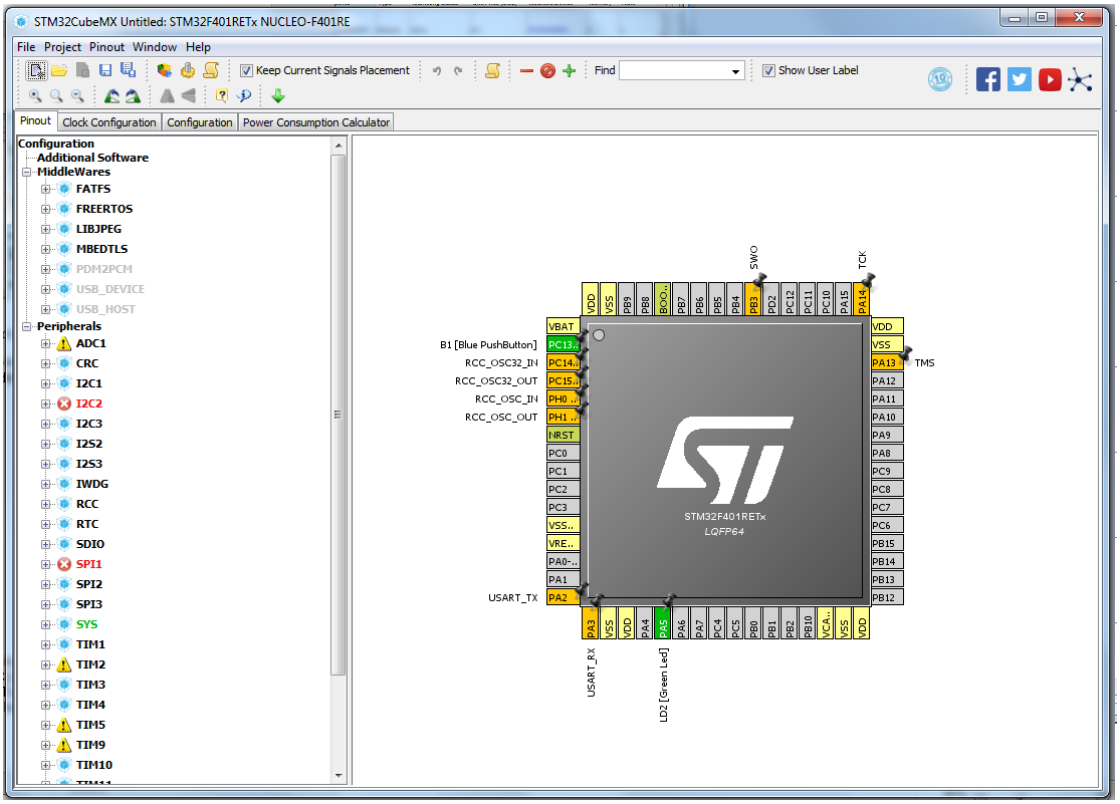

***Figure 3** STM32CubeMX **Configuration** window*

To add the BlueNRG-MS additional software to the project, the button  must be clicked. From the **Additional Software Component Selection** window, the user can either chose to generate, for the selected MCU/Board, one of the enclosed sample applications or a new project. In this latter case, the user must just implement the main application logic without bothering with the pinout and peripherals configuration code which will be automatically

generated by CubeMX.

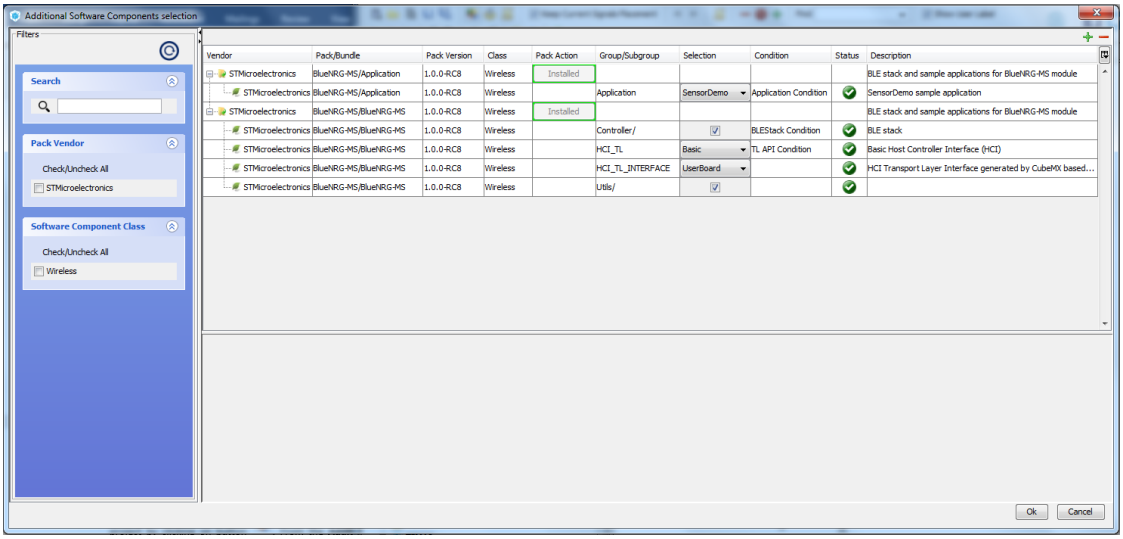

***Figure 4** STM32CubeMX **Additional Software Component Selection** window*

6 STM32 Configuration Steps

The X-NUCLEO-IDB05A1 interfaces with the STM32 microcontroller via the SPI pin. Hence, assuming a user wants to interface the ST X-NUCLEO-IDB05A1 expansion board with a STM32 Nucleo 64 pins board (e.g. a Nucleo-F401RETx) or a STM32 Nucleo 144 pins board (e.g. a Nucleo F429ZITx), the following steps must be executed in CubeMX before generating a project.

***Figure 5** STM32 Nucleo 64 pins and X-NUCLEO-IDB05A1*

If a Nucleo 144 pins is used, to correctly set the SPI clock on pin D13, the D3 pin and the D13 pin of the Arduino connector on the X-NUCLEO-IDB05A1 expansion board must be bridged (alternatively the resistor R9 must be open and a 0 Ohm resistor must be soldiered on R6).

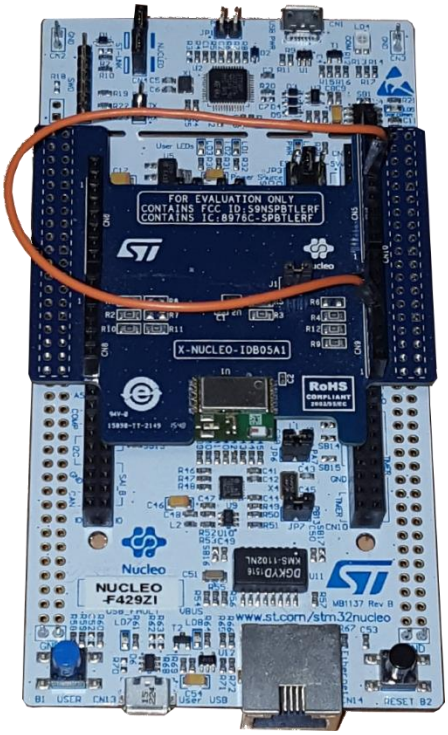

Figure 6 *STM32 Nucleo 144 pins and X-NUCLEO-IDB05A1*

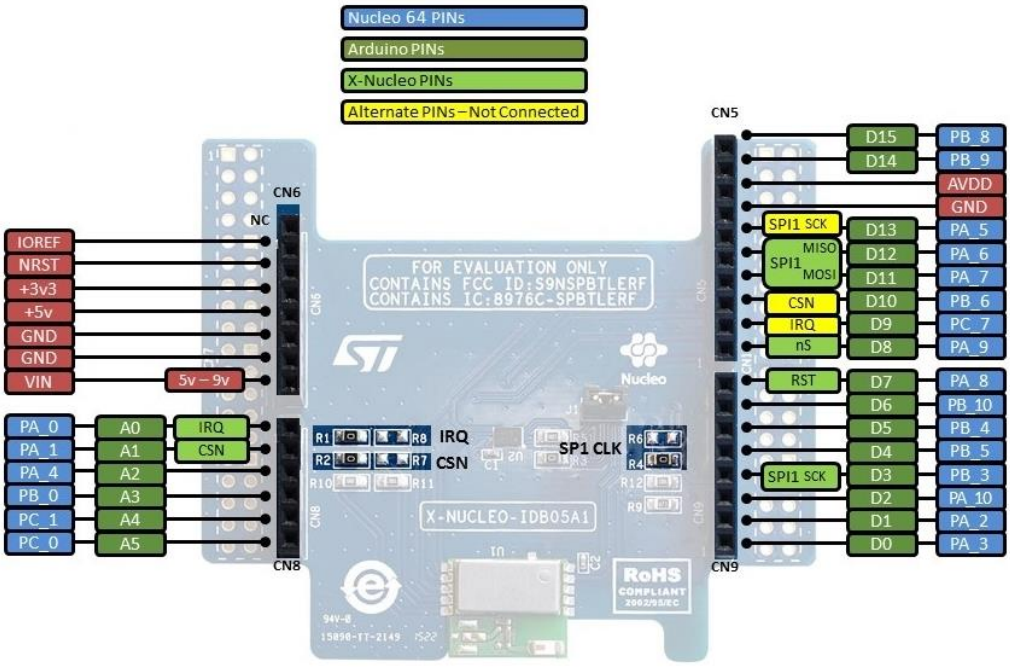

Figure 7 *X-NUCLEO-IDB05A1 pinout*

6.1 Use of Expansion Software without sample applications

This section outlines how to configure CubeMX when the use of the sample applications is not required. With such setup, only middleware and driver layers will be configured. This setup is useful when the user does not intend to leverage the sample application provided in the package.

From the **Peripherals** list:

- from the **Pinout** scheme, if PB3 pin is already assigned, click on it and reset its state;
- from the **Pinout** scheme, reset the state of all pins used by the ETH (LAN8742A) otherwise click on *Pinout* → *Clear Pinouts* (only for Nucleo 144);
- check that the ETH is disabled (only for Nucleo 144);
- enable the SPI1 in Full-Duplex Master Mode;
- if not enabled yet:
    - enable the USART2 in Asynchronous mode (for Nucleo 64)
    - enable the USART3 in Asynchronous mode (for Nucleo 144).

From the **Pinout** scheme set:

Nucleo 64		Nucleo 144	
PA0	GPIO_EXTI0	PA3	GPIO_EXTI3
PA1	GPIO_Output	PC0	GPIO_Output
PA8	GPIO_Output	PF13	GPIO_Output

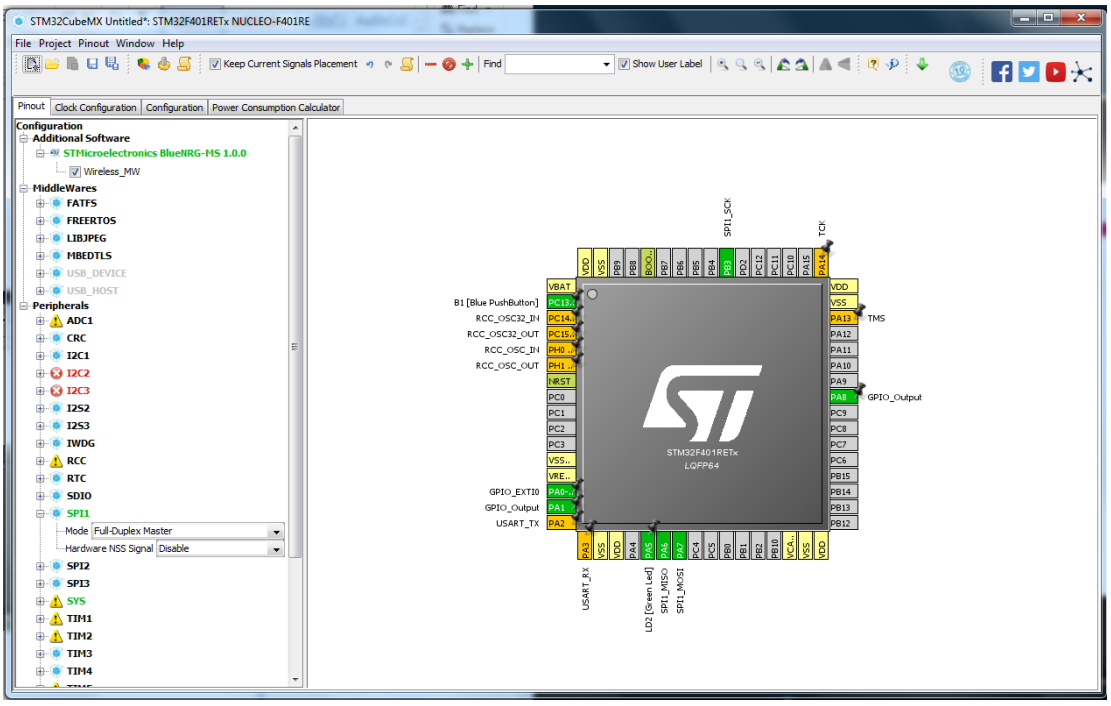

***Figure 8** STM32CubeMX **Pinout** tab*

From the **Configuration** tab, click on BlueNRG-MS buton under the Additional Software and set the following Platform Settings:

Name	BSP_Api	Supported IPs	Nucleo 64	Nucleo 144
BUS IO driver	BSP_BUS_DRIVER	SPI:Full-Duplex Master	SPI1	SPI1

Exti Line	HAL_EXTI_DRIVER	GPIO:EXTI	PA0	PA3
CS Line	Unknown	GPIO:Output	PA1	PC0
Reset Line	Unknown	GPIO:Output	PA8	PF13

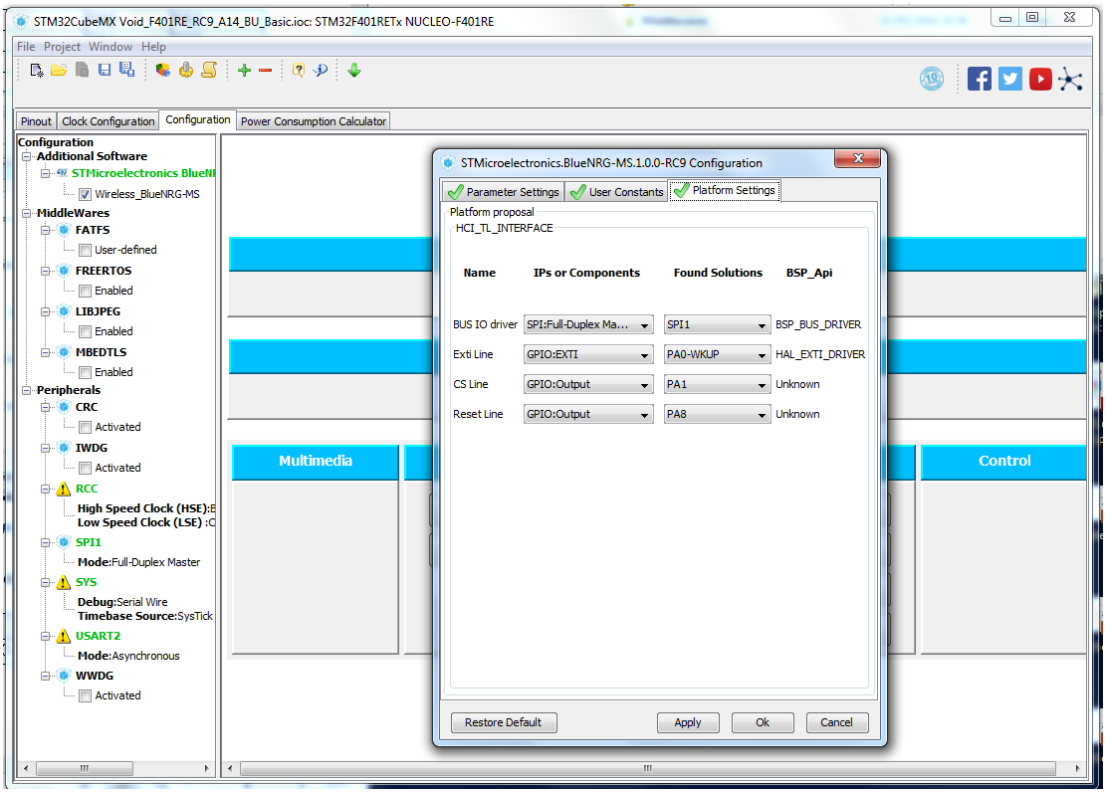

***Figure 9** STM32CubeMX **Configuration** tab*

From the **Configuration** tab, click on NVIC button under System to enable the EXTI line interrupt:

Nucleo 64	Nucleo 144
EXTI line 0 interrupt	EXTI line 3 interrupt

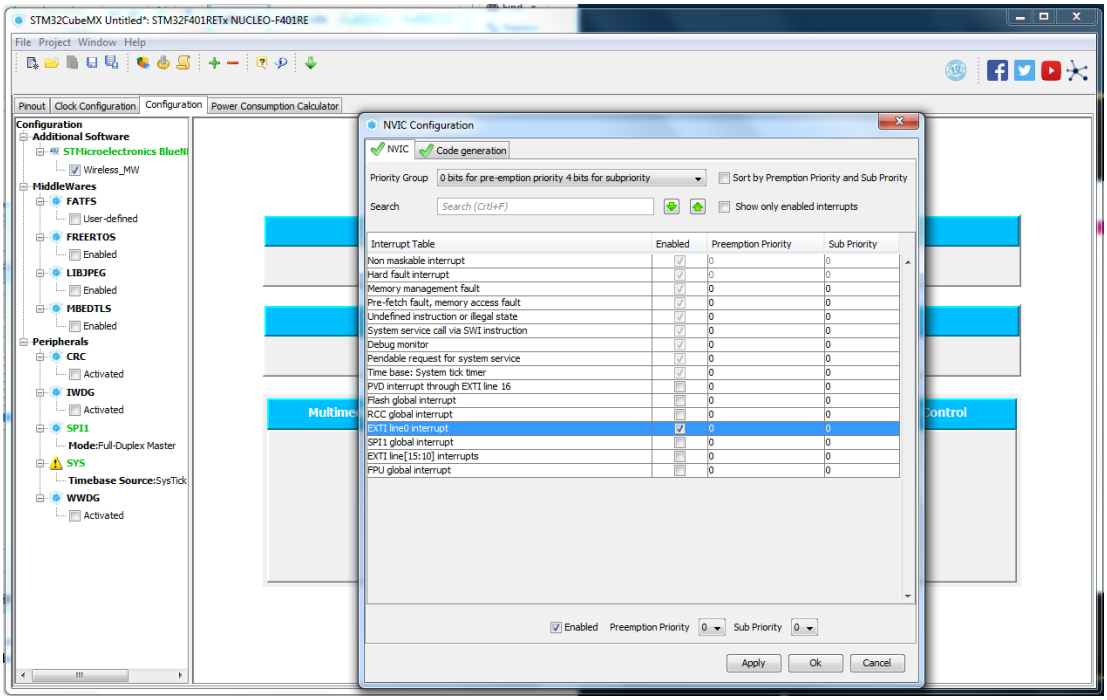

Figure 10** STM32CubeMX **NVIC Configuration

From the **Configuration** tab, click on SPIx button under Connectivity and:

- check that the Data size is 8 Bits;
- set the Prescaler (for Baud Rate) to a value so that HCLK/Prescaler is less or equal to 8MHz (the maximum supported SPI speed)

From **Configuration** tab, click on USARTx button under Connectivity and check that:

- Baud Rate is 115200 Bits/s;
- Word Length is 8 Bits (includeing Parity)
- Parity is None
- Stop Bits is 1

Once all the above described steps have been performed, the source code of the project using the **STMicroelectronics BlueNRG-MS** software can be generated clicking the  button.

In the Advanced Settings tab, check to have the following settings:

Generated Function Calls

Rank	Function Name	IP Instance Name	Not Generate Function Call	Visibility (Static)
1	MX_GPIO_Init	GPIO	☐	☑
2	SystemClock_Config	RCC	☐	☐
3	MX_USART2_UART_Init	USART2	☑	☐
4	MX_BlueNRG_MS_Init	STMicroelectronics.BlueNRG-M...	☐	☑
5	MX_BlueNRG_MS_Process	STMicroelectronics.BlueNRG-M...	☐	☑

***Figure 11** STM32CubeMX Advanced Project settings*

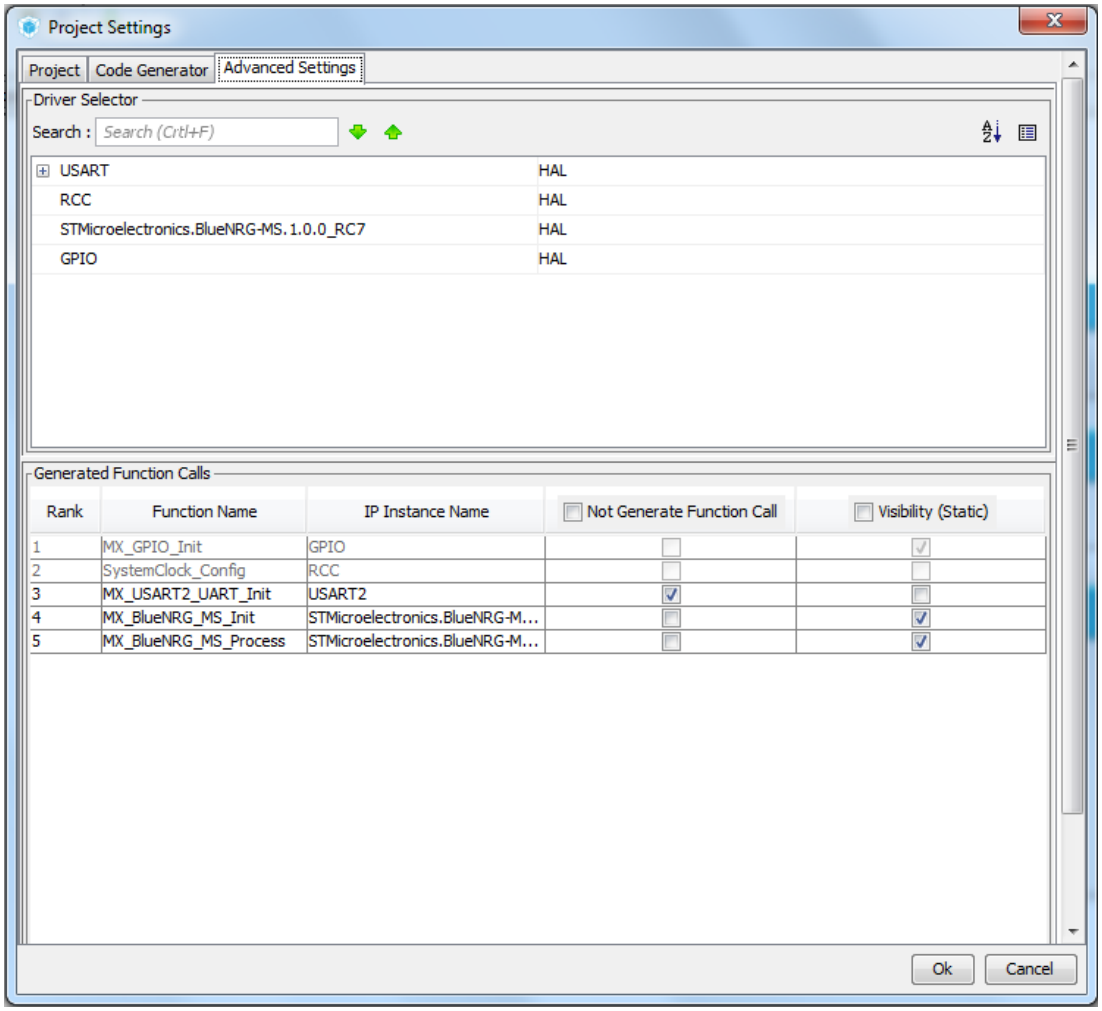

***Figure 12** STM32CubeMX **Advanced Settings** Configuration*

6.2 Use of Expansion Software with sample applications

This section outlines how to configure CubeMX when the use of the sample applications is required. With such setup, all the components of the expansion software package, including applications, will be properly configured.

From the **Peripherals** list:

1. from the **Pinout** scheme, if PB3 pin is already assigned, click on it and reset its state;
2. reset the state of all pins used by the ETH (LAN8742A) otherwise click on *Pinout → Clear Pinouts* (only for Nucleo 144)
3. check that the ETH is disabled (only for Nucleo 144)
4. enable the SPI1 in Full-Duplex Master Mode
5. if not enabled yet:
    a. enable the USART2 in Asynchronous mode (for Nucleo 64)
    b. enable the USART3 in Asynchronous mode (for Nucleo 144)

From the **Pinout** scheme, if not already set, set:

Nucleo 64	Nucleo 144

PIN	Mode	Label	PIN	Mode	Label
PA0	GPIO_EXTI0		PA3	GPIO_EXTI3	
PA1	GPIO_Output		PC0	GPIO_Output	
PA8	GPIO_Output		PF13	GPIO_Output	
PA2	USART2_TX	USART_TX	PD8	USART3_TX	USART_TX
PA3	USART2_RX	USART_RX	PD9	USART3_RX	USART_RX
PA5	GPIO_Output	LD2 [Green Led]	PB7	GPIO_Output	LD2[Blue]
PC13	GPIO_EXTI13	B1 [Blue PushButton]	PC13	GPIO_EXTI13	USER_Btn[B1]

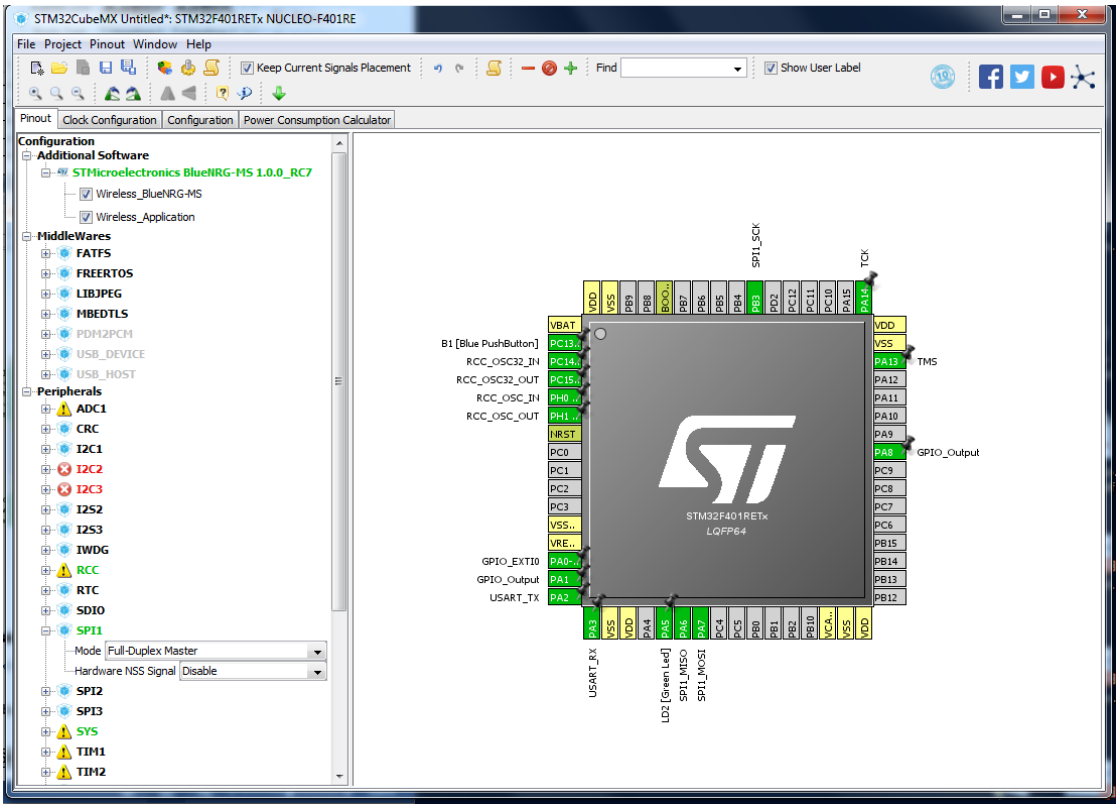

***Figure 13** STM32CubeMX **Pinout** tab*

From the **Configuration** tab, click on BlueNRG-MS button under the Additional Software and set the following Platform Settings:

Name	BSP_Api	Supported IPs	Nucleo 64	Nucleo 144
BUS IO driver	BSP_BUS_DRIVER	SPI:Full-Duplex Master	SPI1	SPI1
Exti Line	HAL_EXTI_DRIVER	GPIO:EXTI	PA0	PA3
CS Line	Unknown	GPIO:Output	PA1	PC0
Reset Line	Unknown	GPIO:Output	PA8	PF13
BSP LED	BSP_COMMON_DRIVER	GPIO:Output	PA5	PB7
BSP BUTTON	BSP_COMMON_DRIVER	GPIO:EXTI	PC13	PC13
BSP USART	BSP_COMMON_DRIVER	USART:Asynchronous	USART2	USART3

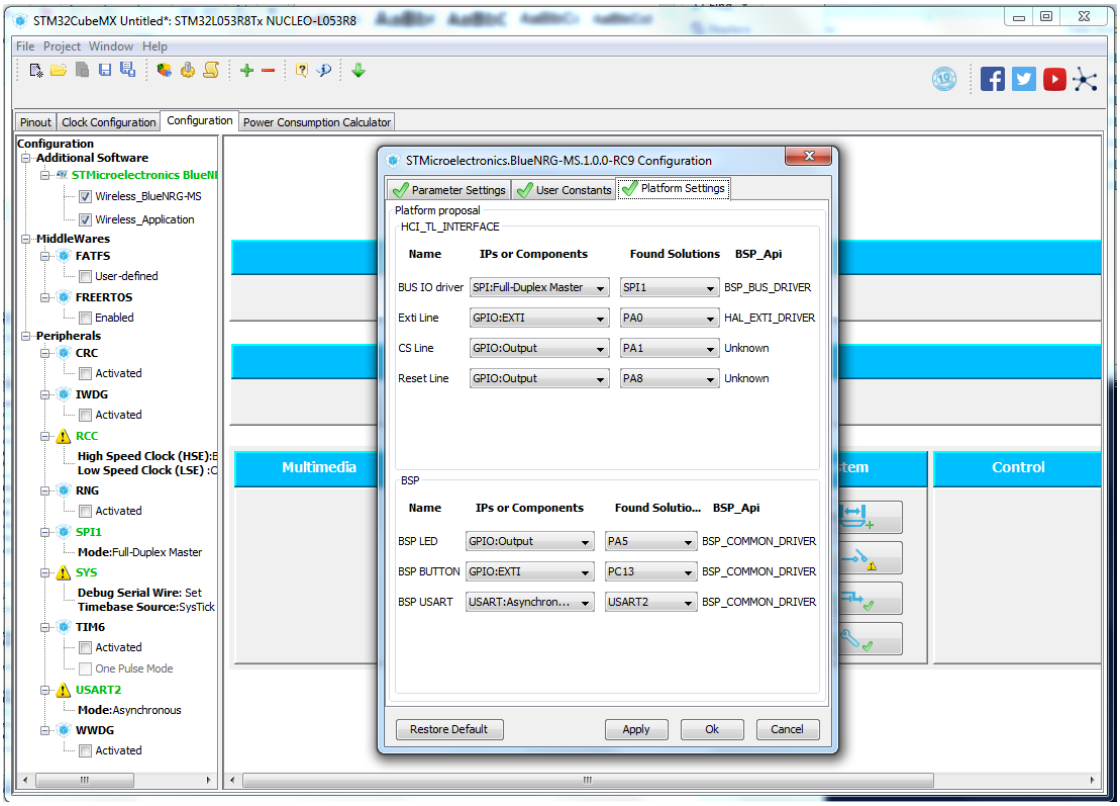

***Figure 14** STM32CubeMX **Configuration** tab*

From the **Configuration** tab, click on NVIC button under System to enable the EXTI line interrupts:

Nucleo 64	Nucleo 144
EXTI line 0 interrupt	EXTI line 3 interrupt

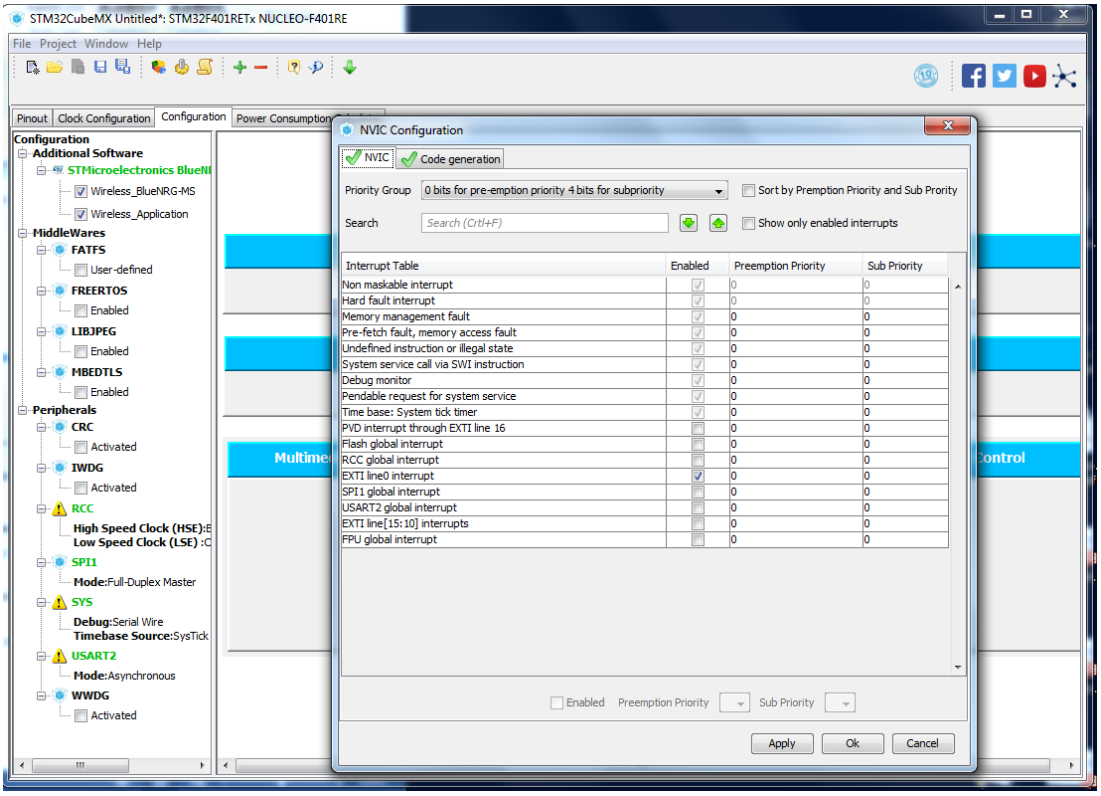

Figure 15 *STM32CubeMX NVIC Configuration*

From the **Configuration** tab, click on SPI1 button under Connectivity and:

- check that the Data Size is 8 Bits;
- set the Prescaler (for Baud Rate) to a value so that HCLK/Prescaler is less or equal to 8MHz (the maximum supported SPI speed).

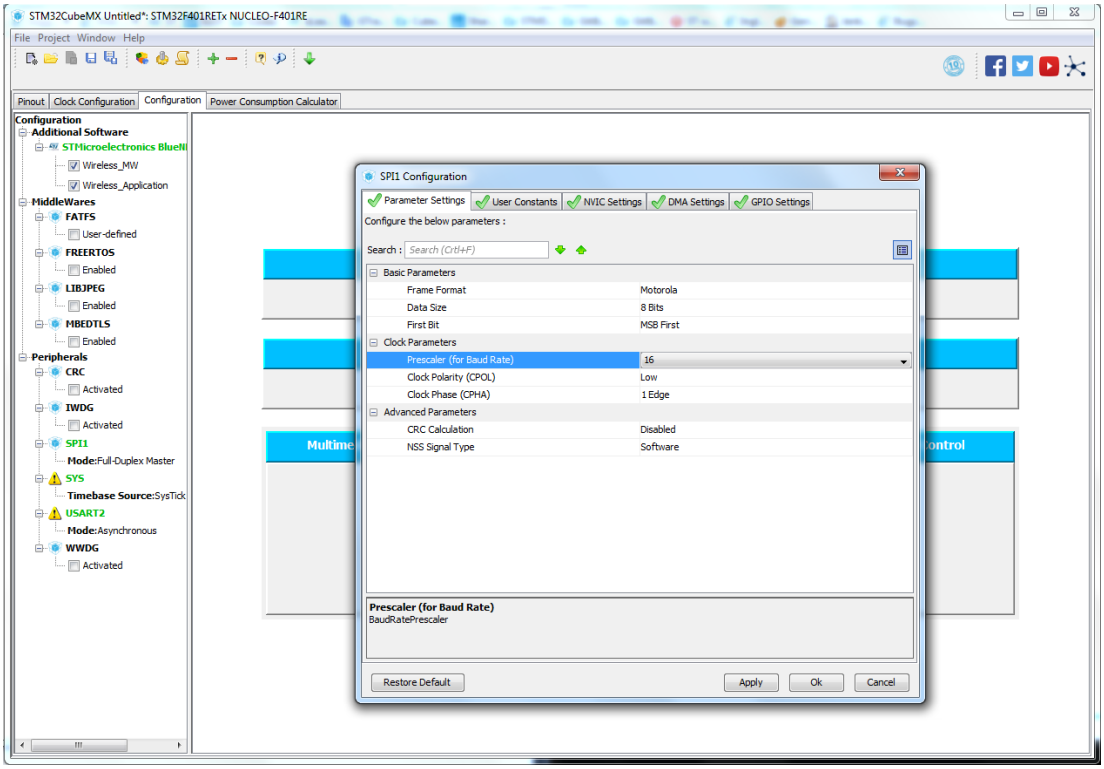

Figure 16 *STM32CubeMX SPI Configuration*

From the Configuration tab, click on USART2 button under Connectivity and check the following configuration is set:

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

Also, from the GPIO Settings tab, be sure the USART_TX and USART_RX label are set.

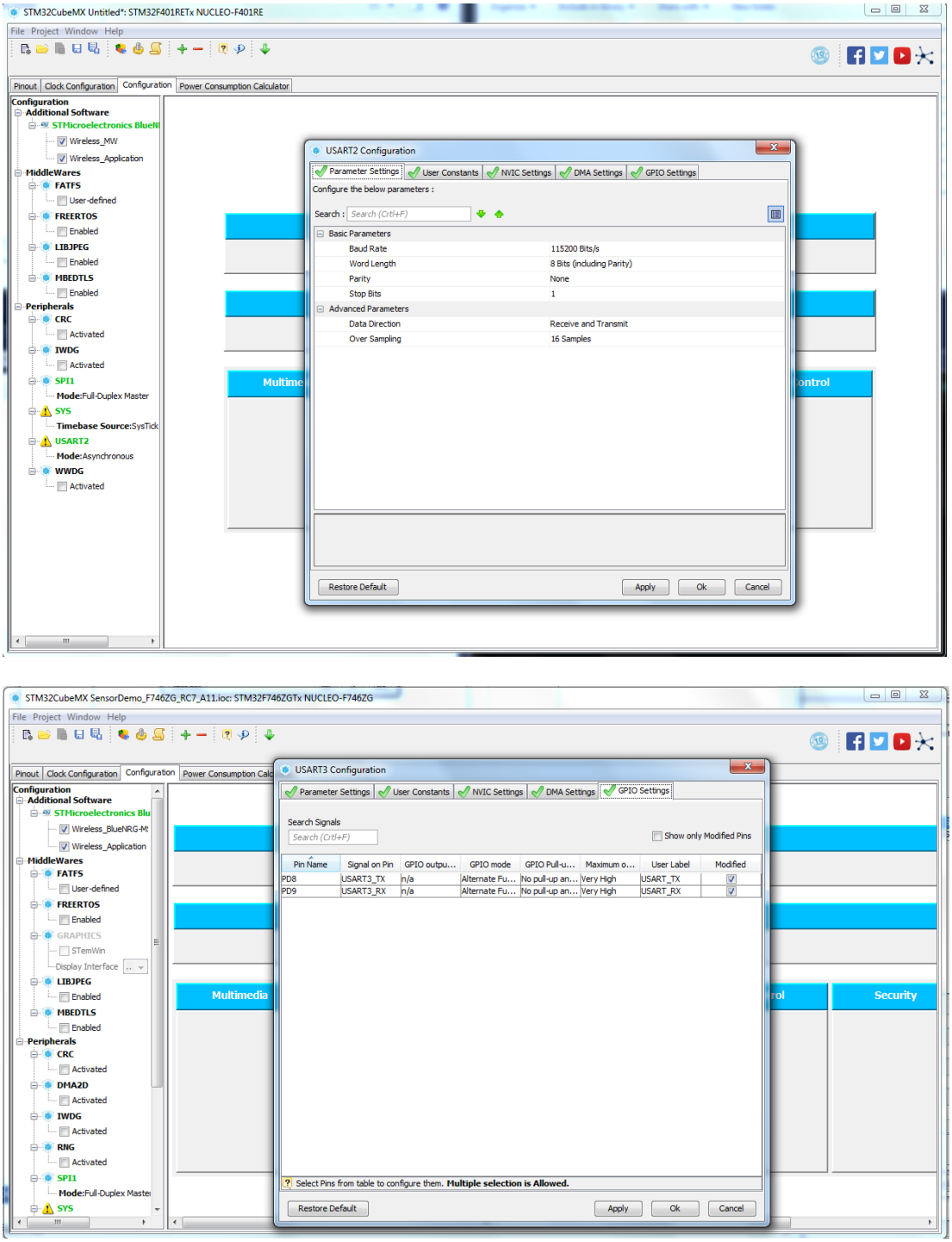

Figure 17 *STM32CubeMX USART Configuration*

Once all the above described steps have been performed, the sample application using the **STMicroelectronics BlueNRG-MS** software can be generated clicking the button.

In the Advanced Settings tab, it is important to replicate the following settings:

Function Name	Not Generate Function Call	Visibility (Static)
MX_USARTX_Init	√	
MX_BlueNRG_MS_Init		√
MX_BlueNRG_MS_Process		√

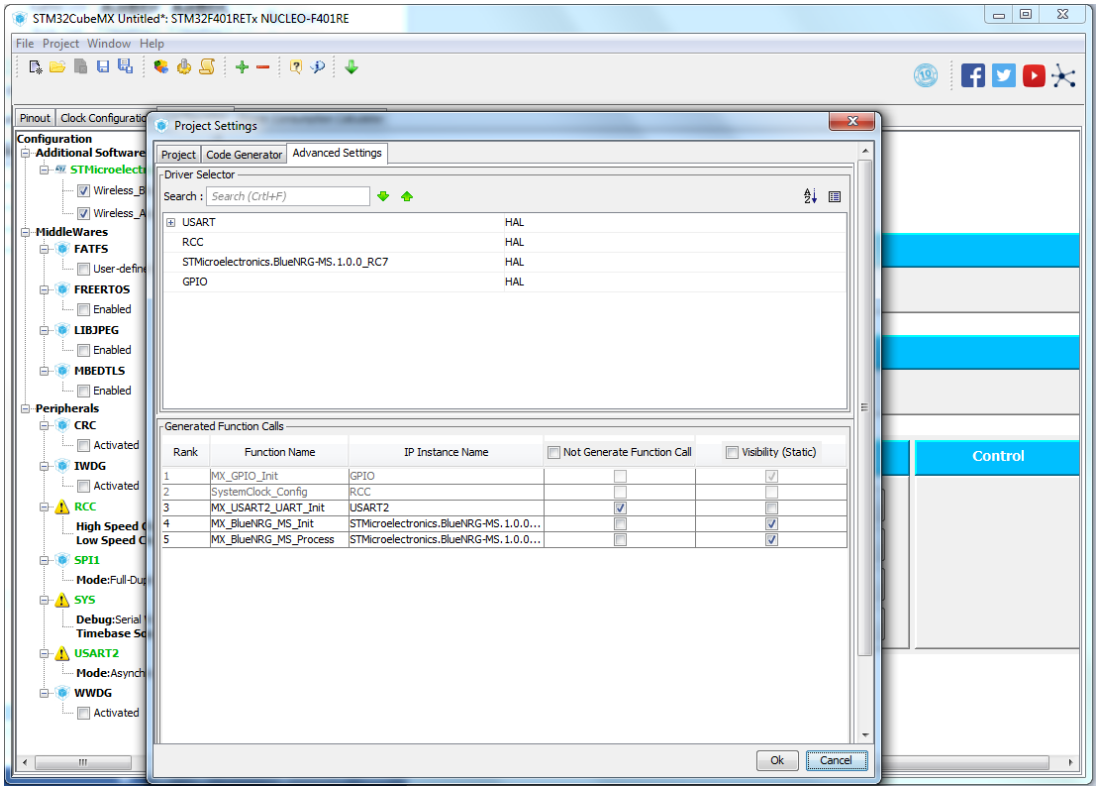

***Figure 18** STM32CubeMX **Advanced Settings** Configuration*

7 Generated Folders Structure

When generating a project, two models of folders structure can be adopted when using a high level firmware component (i.e. a middleware in the STM32Cube MCU package):

- **Basic Structure**: the basic structure is often used with HAL examples and single middleware projects. This structure consists of having the IDE configuration folder in the same level as the sources (organized in *Inc* and *Src* subfolders).
- **Advanced Structure**: the advanced structure provides a more efficient and organized folders model that allows ease middleware applications integration when several middlewares are used.
  In the Advanced mode *Src* and *Inc* are generated under folder *Core*.
  For each middleware, the list of the generated files is under <*MW_Name*> (*BlueNRG-MS* for the BlueNRG-MS middleware), at the same level as *Core* and either in *App* or in *Target* subfolder.

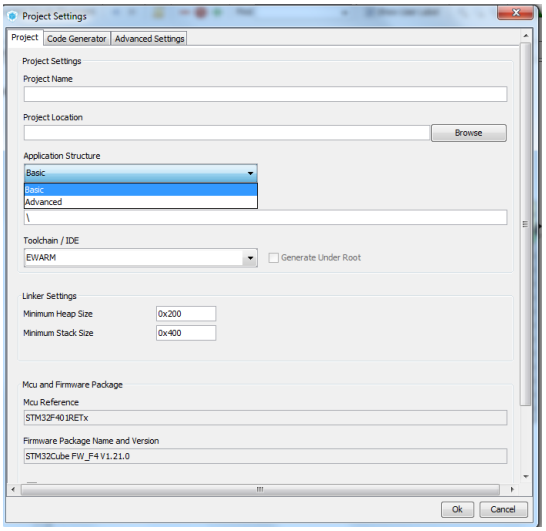

***Figure 19** STM32CubeMX **Application Structure** Configuration*

8 Known Limitations and workarounds

- For sample applications using any low power feature, such as **Beacon**, the ST-Link reset must be set in *Connect during reset* mode into the generated project configuration options.
- All sample applications must be used with the following configuration for the HCI Transoprt Layer (HCI_TL) and the HCI Transport Layer Interface (HCI_TL_INTERFACE):
  - HCI_TL → Basic
  - HCI_TL_INTERFACE → UserBoard

  Configurations using the template files are not supported yet.
- No support to **Low Level (LL) Driver** is provided yet for the SPI interface used by the BlueNRG-MS.

9 References

[1] UM1873 – User Manual - *Getting started with the X-CUBE-BLE1 Bluetooth Low Energy software expansion for STM32Cube* (see section 3.4 "Guide for writing applications")
[2] AN4642 – Application Notes – *Overview of the BLE Profiles application for X-CUBE-BLE1, expansion for STM32Cube*
[3] AN4979 – Application Notes – *Bluetooth Low Energy beacons with Eddystone*

10 Revision history

Table 2: Document revision history

Date	Version	Changes
15-Dec-2017	1	Initial release.
17-Apr-2018	2	Remove Central and Peripheral Profiles sample applications

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.