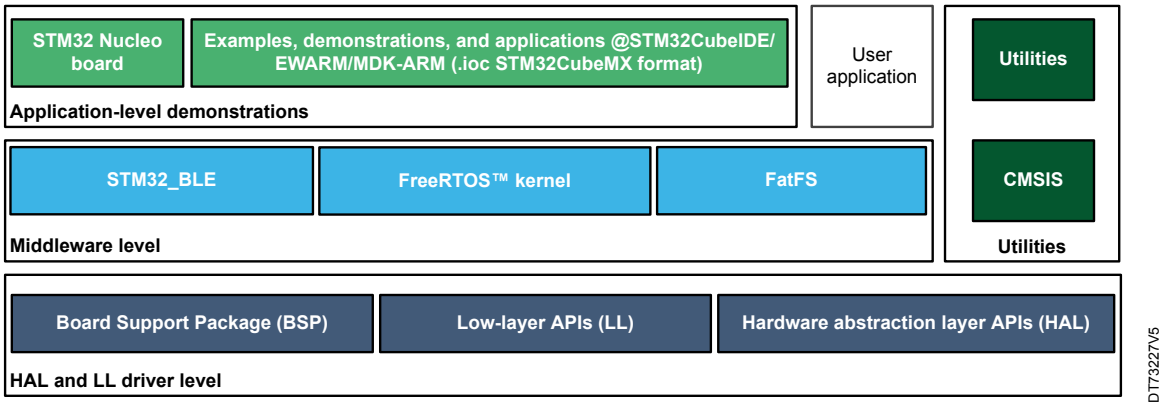
Introduction to STM32Cube MCU Package examples for STM32WB0 MCUs

Introduction

The STM32CubeWB0 MCU Package comes with a rich set of examples running on STMicroelectronics boards. The examples can be found organized in the tables below. They are provided with preconfigured projects for the main supported toolchains (see figure below).

Figure 1. STM32CubeWB0 firmware components



1 General information

This document applies to Arm®-based devices.

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.1 Reference documents

- Latest release of the STM32CubeWB0 firmware package
- *Getting started with STM32CubeWB0 for STM32WB0 series* (UM3205)

2 STM32CubeWB0 examples

The examples are classified depending on the STM32Cube level that they apply to. They are named as follows:

- **Examples**
  The examples use only the HAL and BSP drivers. Middleware components are not used. Their objective is to demonstrate the product/peripherals features and usage. They are organized per peripheral (one folder per peripheral, for example, TIM). Their complexity level ranges go from the basic usage of a given peripheral (for example, PWM generation using timer) to the integration of several peripherals. The usage of the board resources is reduced to the strict minimum.
- **Examples_LL**
  These examples only use the LL drivers. HAL drivers and middleware components are not used. They offer an optimum implementation of typical use cases of the peripheral features and configuration sequences. The LL examples are organized per peripheral (one folder for each peripheral, for example, TIM) and run exclusively on the Nucleo board.
- **Examples_MIX**
  These examples only use HAL, BSP, and LL drivers. Middleware components are not used. They aim at demonstrating how to use both HAL and LL APIs in the same application to combine the advantages of both APIs:
  - HAL drivers offer high-level function-oriented APIs, which have a high level of portability since they hide product/IP complexity to end-users.
  - LL drivers offer low-level APIs at register level with better optimization.

  The examples are organized per peripheral (one folder for each peripheral, for example, TIM) and run exclusively on the Nucleo board.
- **Applications**
  The applications demonstrate the product performance and how to use the available middleware stacks. They are organized either by middleware (one folder per middleware, for example USB host) or by product feature that requires high-level firmware bricks (for example, audio). The integration of applications that use several middleware stacks is also supported.
- **Demonstrations**
  The demonstrations aim at integrating and running the maximum number of peripherals and middleware stacks to showcase the product features and performance.
- **Template project**
  The template project is provided to allow the user to quickly build a firmware application using HAL and BSP drivers on a given board.

The examples are located under *STM32Cube_FW_WB0_VX.Y.Z\Projects*\. they all have the same structure:

- *Inc* folder, containing all header files.
- \Src folder, containing the source code.
- *EWARM*, *MDK-ARM*, and *STM32CubeIDE* folders, containing the preconfigured project for each toolchain.
- readme.txt file, describing the example behavior and the environment required to run the example.

To run the example, proceed as follows:

1. Open the example using your preferred toolchain.
2. Rebuild all files and load the image into target memory.
3. Run the example by following the readme file instructions.

Note: Refer to the "Development toolchains and compilers" and "Supported devices and evaluation boards" sections of the firmware package release notes to know more about the software/hardware environment used for the MCU package development and validation. The correct operation of the provided examples is not guaranteed in other environments, for example when using different compiler or board versions.

The examples can be tailored to run on any compatible hardware: simply update the BSP drivers for your board, provided it has the same hardware functions (LED, push-buttons, etc.). The BSP is based on a modular architecture that can be easily ported to any hardware by implementing the low-level routines.

Table 1. STM32CubeWB0 firmware examples contains the list of examples, demonstrations, and applications provided with STM32WB0 MCU Package.

Note: STM32CubeMX-generated examples are highlighted with the STM32CubeMX icon. Reference materials are available on http://www.st.com/stm32cubefw.

Table 1. STM32CubeWB0 firmware examples

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Templates	-	Starter project	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application.	MX	MX	MX
	Total number of templates: 3			**1**	**1**	**1**
Templates_LL	-	Starter project	This project provides a reference template based on the STM32Cube LL API that can be used to build any firmware application.	MX	MX	MX
	Total number of templates_ll: 3			**1**	**1**	**1**
Examples	ADC	ADC_AnalogWatchdog	How to use an ADC peripheral with an ADC analog watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds.	MX	MX	MX
		ADC_MultiChannelSingleConversion	How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence.	MX	-	-
		ADC_Downsampling	How to use an ADC peripheral with downsampling	MX	MX	MX
	CORTEX	CORTEXM_MPU	This example presents the MPU features. It configures the MPU attributes of different MPU regions. Then, it configures a memory area as privileged read-only, and attempts to perform read and write operations in different modes.	MX	-	-
		CORTEXM_ModePrivilege	How to modify the Thread mode privilege access and stack. Thread mode is entered on reset or when returning from an exception.	MX	-	-
	CRC	CRC_Data_Reversing_16bit_CRC	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes a 16-bit CRC code derived from a buffer of 32-bit data (words). Input and output data reversal features are enabled. The user-defined generating polynomial is manually set to 0x1021, that is, X^16 + X^12 + X^5 + 1, which is the CRC-CCITT generating polynomial.	MX	-	-
		CRC_Example	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the CRC code of a given buffer of 32-bit data words, using a fixed generator polynomial (0x4C11DB7).	MX	-	-
		CRC_UserDefinedPolynomial	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the 8-bit CRC code for a given buffer of 32-bit data words, based on a user-defined generating polynomial.	MX	MX	MX
	DMA	DMA_RAMToRAM	How to use a DMA to transfer a word data buffer from embedded SRAM to embedded SRAM through the HAL API.	MX	-	-

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Examples	FLASH	FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal flash memory. At the beginning of the main program, the HAL_Init() function is called to reset all the peripherals, initialize the flash memory interface and the SysTick.	MX	MX	MX
		FLASH_WriteProtection	How to configure and use the FLASH HAL API to enable and disable the write protection of the internal flash memory.	MX	-	-
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	MX	MX	MX
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	MX	-	-
	HAL	HAL_TimeBase	How to customize HAL using a general-purpose timer as the main source of time base instead of the SysTick.	MX	-	-
		HAL_TimeBase_RTC_WKUP	How to customize HAL using RTC wake-up as the main source of time base instead of SysTick.	MX	-	-
		HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as the main source of time base instead of SysTick.	MX	MX	MX
	I²C	I2C_TwoBoards_AdvComIT	How to handle several I2C data buffer transmission/reception between a master and a slave device, using an interrupt.	MX	MX	MX
		I2C_TwoBoards_ComDMA	How to handle I2C data buffer transmission/ reception between two boards, via DMA.	MX	-	-
		I2C_TwoBoards_ComIT	How to handle I2C data buffer transmission/ reception between two boards, using an interrupt.	MX	-	-
		I2C_TwoBoards_ComPolling	How to handle I2C data buffer transmission/ reception between two boards, in polling mode.	MX	-	-
	IWDG	IWDG_Reset	How to handle the IWDG reload counter and simulate a software fault that generates an IWDG reset after a preset lap of time.	MX	-	-
		IWDG_WindowMode	How to periodically update the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset laps of time.	MX	MX	MX
	PKA	PKA_ECDSA_Sign	How to compute a signed message regarding the elliptic curve digital signature algorithm (ECDSA).	MX	-	MX
		PKA_ECDSA_Verify	How to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA).	MX	-	-
		PKA_ModularExponentiation	How to use the PKA peripheral to execute modular exponentiation. This allows ciphering/deciphering a text.	MX	-	-

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Examples	PWR	PWR_DEEPSTOP	How to enter the Deepstop mode and wake up from this mode by using an external reset or the WKUP pin.	MX	MX	MX
		PWR_PVD	How to use the PVD feature.	MX	-	-
		PWR_DEEPSTOP_RTC	How to enter the Deepstop mode and wake-up from this mode by using an external reset or the RTC wake-up timer.	MX	-	-
	RADIO	RADIO_Beep	This code implements a transmission only example.	MX	MX	MX
		RADIO_Beep_Encrypted	This code implements a transmission only example with encryption enabled.	MX	MX	MX
		RADIO_BeepMultiState	Shows how to configure a multiple-state machine for transmission on different channels using the ActionPacket mechanism exported by the 2.4 GHz radio driver.	MX	MX	MX
		RADIO_RemoteControl	This code implements a basic remote control scenario.	MX	MX	MX
		RADIO_SerialPort	This code implements a point to point two-way communication. Two devices are necessary to run fully this demo.	MX	MX	MX
		RADIO_SerialPort_Encrypted	This code implements a point to point two-way encrypted communication. Two devices are necessary to run fully this demo.	MX	MX	MX
		RADIO_Skeleton	Code demonstrating the basic project structure template with initialization framework to be used for building a 2.4 GHx radio example application.	MX	MX	MX
		RADIO_Sniffer	Code demonstrating a sniffer application on a specific frequency channel.	MX	MX	MX
		RADIO_Sniffer_Encrypted	Code demonstrating a sniffer application on a specific frequency channel with encryption enabled.	MX	MX	MX
		RADIO_SnifferMultiState	Shows how to configure receiving packets on different channels using multiple-state machines.	MX	MX	MX
		RADIO_StarNetwork_Central	This shows how to implement a star network using 2.4 GHz radio proprietary driver. At least two devices are needed. This application implements the central device role.	MX	MX	MX
		RADIO_StarNetwork_Peripheral	This shows how to implement a star network using a 2.4 GHz radio proprietary driver. This application implements the peripheral device role.	MX	MX	MX
		RADIO_TestRx	Code demonstrating a simple TX/RX scenario. This code implements the receiver side.	MX	MX	MX

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Examples	RADIO	RADIO_TestTx	Code demonstrating a simple TX/RX scenario. This code implements the transmitter side.	MX	MX	MX
		RADIO_Throughput_RXB	This code implements a throughput test (receiver configuration).	MX	MX	MX
		RADIO_Throughput_TX	This code implements a throughput test (transmitter, unidirectional configuration: only one device is needed).	MX	MX	MX
		RADIO_Throughput_TXB	This code implements a throughput test (transmitter, bidirectional).	MX	MX	MX
	RCC	RCC_ClockConfig	Configuration of the system clock (SYSCLK) and modification of the clock settings in Run mode, using the RCC HAL API.	MX	MX	MX
		RCC_LSEConfig	Enabling/disabling of the low-speed external(LSE) RC oscillator (about 32 kHz) at runtime, using the RCC HAL API.	-	MX	-
		RCC_LSIConfig	How to enable/disable the low-speed internal (LSI) RC oscillator (about 32 kHz) at runtime, using the RCC HAL API.	-	MX	-
	RNG	RNG_MultiRNG	Configuration of the RNG using the HAL API. This example uses the RNG to generate 32-bit long random numbers.	MX	MX	MX
	RTC	RTC_Alarm	Configuration and generation of an RTC alarm using the RTC HAL API.	MX	MX	MX
		RTC_Calendar	Configuration of the calendar using the RTC HAL API.	MX	-	-
	SPI	SPI_FullDuplex_ComDMA_Master	Data buffer transmission/reception between two boards via SPI using DMA.	MX	MX	MX
		SPI_FullDuplex_ComDMA_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	MX	MX	MX
		SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI using Polling mode.	MX	-	-
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI using Polling mode.	MX	-	-
	TIM	TIM_InputCapture	How to use the TIM peripheral to measure an external signal frequency.	MX	-	-
		TIM_TimeBase	This example shows how to configure the TIM peripheral to generate a time base of one second with the corresponding Interrupt request.	MX	MX	MX
	UART	LPUART_TwoBoards_ComIT	LPUART transmission (transmit/receive) in Interrupt mode between two boards.	MX	MX	MX
		UART_HyperTerminal_DMA	UART transmission (transmit/receive) in DMA mode between a board and a HyperTerminal PC application.	MX	MX	MX

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Examples	UART	UART_HyperTerminal_IT	UART transmission (transmit/receive) in Interrupt mode between a board and a HyperTerminal PC application.	MX	-	-
		UART_Printf	Rerouting of the C library printf function to the UART.	MX	-	-
	Total number of examples: 130			**59**	**36**	**35**
Examples_LL	CORTEX	CORTEX_MPU	Presentation of the MPU features. This example configures the MPU attributes of different MPU regions then configures a memory area as privileged read-only, and attempts to perform read and write operations in different modes.	MX	MX	MX
	GPIO	GPIO_InfiniteLedToggling_Init	How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32WB0 LL API. The peripheral is initialized with the LL initialization function to demonstrate LL init usage.	MX	-	-
	IWDG	IWDG_RefreshUntilUserEvent_Init	How to configure the IWDG peripheral to ensure periodical counter update and generate an MCU IWDG reset when a USER push-button is pressed. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	MX	-	-
	LPUART	LPUART_WakeUpFromDeepStop_Init	Configuration of GPIO and LPUART peripherals to allow characters received on the LPUART_RX pin to wake up the MCU from low-power mode. This example is based on the LPUART LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	MX	-	-
	PKA	PKA_ECDSA_Sign	How to use the low-layer PKA API to generate an ECDSA signature.	MX	-	MX
	RNG	RNG_GenerateRandomNumbers	Configuration of the RNG to generate 32-bit long random numbers. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	-	-
	RTC	RTC_Alarm_Init	Configuration of the RTC LL API to configure and generate an alarm using the RTC peripheral. The peripheral initialization uses the LL initialization function.	MX	-	-
	TIM	TIM_PWMOutput_Init	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32WB0x TIM LL API. The peripheral initialization uses the LL initialization function to demonstrate LL Init.	MX	-	-

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Examples_LL	USART	USART_Communication_Rx_IT_ Continuous_Init	This example shows how to configure the GPIO and USART peripherals for continuously receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	MX	MX	MX
		USART_Communication_Tx_IT_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on the STM32WB0x USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	MX	MX	MX
	UTILS	UTILS_ConfigureSystemClock	Use of UTILS LL API to configure the system clock using PLL with HSI as source clock.	MX	MX	MX
		UTILS_ReadDeviceInfo	This example reads the UID, the Device ID, and the Revision ID and saves them into a global information buffer.	MX	-	-
	Total number of examples_ll: 21			**12**	**4**	**5**
Applications	Bluetooth® Low Energy	BLE_ApplicationInstallManager	The BLE_ApplicationInstallManager application, associated to a Bluetooth® Low Energy application embedding OTA service, manages the firmware update over the air of a Bluetooth® Low Energy application.	MX	MX	-
		BLE_ANCS_CentralPeripheral	The ANCS (Apple Notification Center Service) demo configures a STM32WB0x device as a Notification Consumer.	MX	MX	MX
		BLE_Beacon	How to advertise 4 types of beacon (tlm, uuid, url, iBeacon).	MX	MX	MX
		BLE_Beacon_AoA_Tag	This allows the configuration of an STM32CubeWB0 device as an AoA tag for a connectionless scenario. The device advertises beacon packets containing Constant Tone Extension data.	MX	-	MX
		BLE_Beacon_Scanner_AoA_Locator	This shows how to receive advertising beacon packets containing Constant Tone Extension data and collect the IQ samples.	MX	-	MX
		BLE_DataThroughput_Client	How to demonstrate point-to-point communication using Bluetooth® Low Energy component as a GATT server or a GATT client.	MX	MX	MX
		BLE_DataThroughput_Server	How to demonstrate point-to-point communication using Bluetooth® Low Energy component as a GATT server or a GATT client.	MX	MX	MX
		BLE_DirectionFinding_Central_Locator	This demo implements a basic direction finding scenario (locator role & connection mode) to demonstrate how to utilize the related Bluetooth LE stack capabilities.	MX	-	MX

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Applications	Bluetooth® Low Energy	BLE_DirectionFinding_Peripheral_Tag	This demo implements a basic direction finding scenario (tag role, connection CTE mode) to demonstrate how to utilize the related Bluetooth LE stack capabilities.	MX	-	MX
		BLE_HealthThermometer	How to use the health thermometer profile as specified by the Bluetooth® Low Energy SIG.	MX	MX	MX
		BLE_HeartRate	How to use the heart rate profile as specified by the Bluetooth® Low Energy SIG.	MX	MX	MX
		BLE_HeartRate_ota	How to use the heart rate and OTA profile as specified by the Bluetooth® Low Energy SIG.	MX	MX	-
		BLE_MultipleConnections_Central	Demonstrate STM32WB0 acting as a central, GATT client, in a multiple connection scenario.	X	X	X
		BLE_MultipleConnections_ CentralPeripheral	Demonstrate STM32WB0 acting as a central, peripheral and GATT client/server, in a multiple connection scenario.	X	X	X
		BLE_MultipleConnections_Peripheral	Demonstrate STM32WB0 acting as a peripheral, GATT server, in a multiple connection scenario.	X	X	X
		BLE_HID_Keyboard	How to use a human interface device over a GATT profile for a keyboard as specified by the Bluetooth® Low Energy SIG.	MX	MX	MX
		BLE_HID_Mouse	How to use a human interface device over a GATT profile for a mouse as specified by the Bluetooth® Low Energy SIG.	MX	MX	MX
		BLE_PAwR_Broadcaster	Demonstrate a STM32WB0 device acting as a Broadcaster of a Periodic Advertising with Response (PAwR) train.	X	-	-
		BLE_PAwR_Observer	Demonstrate a STM32WB0 device acting as an Observer of a Periodic Advertising with Response (PAwR) train.	X	-	-
		BLE_Peripheral_Lite	Demonstrate how to communicate with simple Bluetooth® Low Energy peripheral with minimum activated features.	MX	MX	MX
		BLE_PowerControl_Central	This example shows the LE Power Control features on Central role (Dynamic adjustment of TX power level on a connection, Path loss monitoring).	MX	MX	MX
		BLE_PowerControl_Peripheral	This example shows the LE Power Control features on Perpheral role (Dynamic adjustment of TX power level on a connection, Path loss monitoring).	MX	MX	MX
		BLE_PowerConsumption	Basic Bluetooth® Low Energy demo to demonstrate the device power consumption in specific scenarios (ADV 100 ms, ADV 1000 ms, and CONNECTION).	MX	MX	MX
		BLE_Privacy_Central	It implements a controller privacy scenario using privacy 1.2 available with STM32WB0 Bluetooth® Low Energy stack v4.x (central role).	MX	MX	MX

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Applications	Bluetooth® Low Energy	BLE_Privacy_Peripheral	It implements a Controller Privacy scenario using Privacy 1.2 available with STM32WB0 Bluetooth® Low Energy stack v4.x (peripheral role).	MX	MX	MX
		BLE_RC_LongRange_Central	This demo shows how to control a remote device (e.g., to drive an actuator) using the STM32WB0 Coded PHY feature to reach longer distances.	MX	MX	MX
		BLE_RC_LongRange_Peripheral	This demo shows how to control a remote device (e.g., to drive an actuator) using the STM32WB0 Coded PHY feature to reach longer distances.	MX	MX	MX
		BLE_Security_Central	This demonstrates STM32WB0 acting as a Bluetooth® Low Energy central and GATT client with security framework.	MX	MX	MX
		BLE_Security_Peripheral	This demonstrates STM32WB0 acting as a Bluetooth® Low Energy peripheral and GATT server with security framework.	MX	MX	MX
		BLE_SerialCom_Central	How to demonstrate point-to-point communication using Bluetooth® Low Energy L2CAP component.	MX	MX	MX
		BLE_SerialCom_Peripheral	How to demonstrate point-to-point communication using Bluetooth® Low Energy L2CAP component.	MX	MX	MX
		BLE_SerialPort_CentralPeripheral	Demonstrate point-to-point communication with Bluetooth Low Energy (BLE) GATT custom profile. It operates as a GAP central and peripheral device.	MX	MX	MX
		BLE_SerialPort_Client	How to demonstrate point-to-point communication using Bluetooth® Low Energy GATT profile.	MX	MX	MX
		BLE_SerialPort_Server	How to demonstrate point-to-point communication using Bluetooth® Low Energy GATT profile.	MX	MX	MX
		BLE_Skeleton	Bluetooth® Low Energy application skeleton (Bluetooth® Low Energy stack modular configuration and initialization parameters).	MX	MX	MX
		BLE_StaticStack	It implements the STM32WB0 Bluetooth LE static stack image.	X	X	X
		BLE_StaticStack_ota	It implements the STM32WB0 Bluetooth LE static stack with over-the-air capability image.	X	X	-
		BLE_TransparentMode	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor (UART mode).	MX	MX	MX
		BLE_TransparentMode_SPI	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor (SPI mode).	X	X	X

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Applications	Bluetooth® Low Energy	BLE_TransparentMode_SPI_C_O	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor controller only (SPI mode).	X	X	X
		BLE_TransparentMode_SPI_for_Updater	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor (SPI mode for Updater).	X	X	X
		BLE_TransparentMode_SPI_for_Updater_C_O	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor controller only (SPI mode for Updater).	-	-	X
		BLE_TransparentMode_SPI_Updater	Updater application for updating the BLE_TransparentMode application built for supporting the updater (SPI mode).	X	X	X
		BLE_TransparentMode_C_O	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor controller only (UART mode).	MX	MX	MX
		BLE_TransparentMode_UART_for_Updater	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor (UART mode for Updater).	X	X	X
		BLE_TransparentMode_UART_for_Updater_C_O	How to use the Bluetooth® Low Energy stack running on an STM32WB0 device configured as a network coprocessor controller only (UART mode for Updater).	-	-	X
		BLE_TransparentMode_UART_Updater	Updater application allowing the update of the BLE_TransparentMode application built for supporting the updater (UART mode)	X	X	X
		BLE_p2pClient	Demonstrate STM32WB0 acting as a Bluetooth® Low Energy central and a GATT client.	MX	MX	MX
		BLE_p2pClient_Ext	Demonstrate a Bluetooth® Low Energy scanner with connections from an extended and legacy advertising.	MX	MX	MX
		BLE_p2pRouter	Demonstrate STM32WB0 acting at the same time as both Bluetooth® Low Energy central and peripheral, GATT server, and client.	MX	MX	MX
		BLE_p2pServer	Demonstrate STM32WB0 acting as Bluetooth® Low Energy peripheral and GATT server.	MX	MX	MX
		BLE_p2pServer_Ext	Demonstrate STM32WB0 acting as Bluetooth® Low Energy peripheral and GATT server and using several advertising sets.	MX	MX	MX
		BLE_p2pServer_FreeRTOS	Demonstrate STM32WB0 acting as Bluetooth® Low Energy peripheral and GATT server with FreeRTOS.	X	X	X
		BLE_p2pServer_StaticStack	BLE_p2pserver application supporting Bluetooth LE static stack.	X	X	X

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Applications	Bluetooth® Low Energy	BLE_p2pServer_StaticStack_ota	BLE_p2pserver application with over-the-air feature supporting Bluetooth LE static stack with over-the-air capability.	X	X	-
		BLE_p2pServer_ota	Demonstrate STM32WB0 acting as Bluetooth® Low Energy peripheral and GATT server and offering an OTA firmware update service.	MX	MX	-
	FreeRTOS	FreeRTOS_Mutex	This application demonstrates the use of mutexes to serialize access to a shared resource.	MX	MX	-
		FreeRTOS_Queues_ThreadFlags	This application demonstrates the use of message queues, and thread flags with CMSIS_RTOS2 API	MX	-	MX
		FreeRTOS_Semaphore_LowPower	This application demonstrates the use of FreeRTOS tickless low-power mode and semaphores	MX	-	-
	Total number of applications: 156			**57**	**49**	**50**
Demonstrations	RADIO	RADIO_Skeleton	Code demonstrating the basic project structure template with initialization framework to be used for building a 2.4 GHz radio demonstration application.	X	X	X
		RADIO_SleepRx	Code demonstrating a simple TX/RX scenario with sleep management. This code implements the receiver side. Two devices are necessary to run fully this demo.	X	X	X
		RADIO_SleepTx	Code demonstrating a simple TX/RX scenario with sleep management. This code implements the transmitter side. Two devices are necessary to run fully this demo.	X	X	X
		RADIO_otaClient	Proprietary over-the-air firmware upgrade application, client role. Two devices are needed for performing an over-the-air firmware upgrade.	X	X	X
		RADIO_otaRx	Code demonstrating a simple TX/RX scenario with over-the-air firewall upgrade capability. This code implements the receiver side. Image is built with proper offset to be used with OTA Client for over-the-air firmware upgrade. Sleep management is also supported.	X	X	X
		RADIO_OTAServerFixed	Proprietary over-the-air firmware upgrade application, Server with fixed image role. Two devices are needed for performing an over-the-air firmware upgrade.	X	X	X
		RADIO_otaServerYmodem	Proprietary over-the-air firmware upgrade application, server with Ymodem role. Two devices are necessary to fully run this demo.	X	X	X

Level	Module name	Project name	Description	STM32WB09 NUCLEO	STM32WB07 NUCLEO	STM32WB05 NUCLEO
Demonstrations	RADIO	RADIO_otaTx	Code demonstrating a simple TX/RX scenario with over-the-air firewall upgrade capability. This code implements the transmitter side. Image is built with proper offset to be used with OTA Client for over-air-firmware upgrade. Sleep management is also supported.	X	X	X
	RADIO_ TIMER	RADIO_TIMER_Counter	Code demonstrating how to use radio timer for triggering periodic wake-up.	X	X	X
		RADIO_TIMER_HSEStartupTime	Code demonstrating how to calculate the HSE startup time required for proper radio timer functionality.	X	X	X
	Total number of demonstrations: 30			**10**	**10**	**10**
Total number of applications: 343				**140**	**101**	**102**

Revision history

Table 2. **Document revision history**

Date	Version	Changes
13-Jun-2024	1	Initial release
04-Nov-2024	2	Updated: • Figure 1. STM32CubeWB0 firmware components • Section 1: General information • Table 1. STM32CubeWB0 firmware examples
17-Feb-2025	3	Updated: Table 1. STM32CubeWB0 firmware examples

Contents

List of tables

List of figures