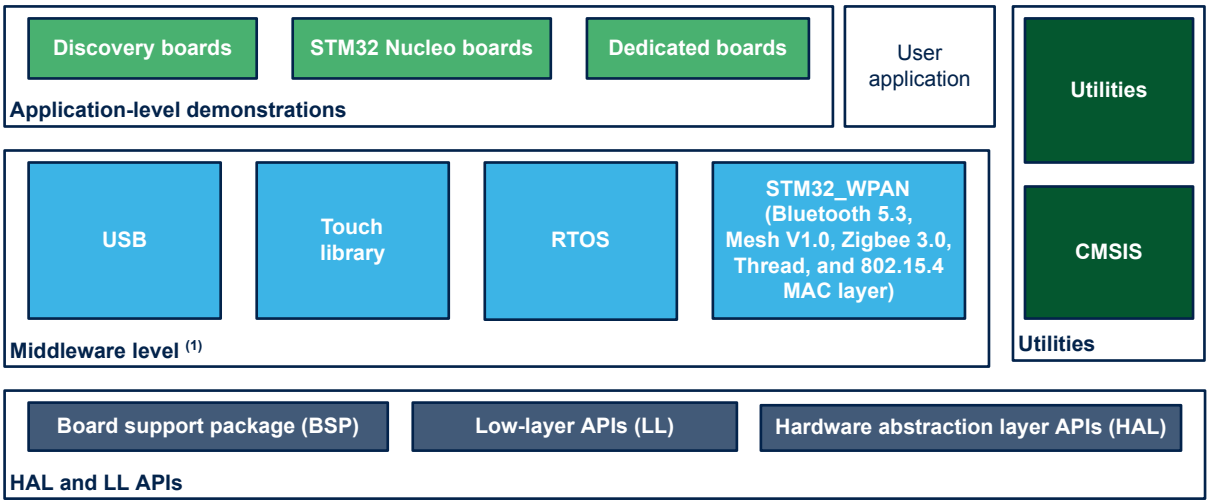
# STM32Cube MCU Package examples for STM32WB Series

## Introduction

The STM32CubeWB MCU Package comes with a rich set of examples running on STMicroelectronics boards. The examples are organized by board, and are provided with preconfigured projects for the main supported toolchains (see figure below).

**Figure 1. STM32CubeWB firmware components**




(1) The set of middleware components depends on the product Series.

# 1 Reference documents

The reference documents are available on www.st.com/stm32cubefw:

- Latest release of STM32CubeWB firmware package
- *Getting started with STM32CubeWB for STM32WB Series* (UM2550)
- *STM32CubeWB Nucleo demonstration firmware* (UM2551)
- *Description of STM32WB HAL and low-layer drivers* (UM2442)
- *Developing applications on STM32Cube with FatFS* (UM1721)
- *Developing applications on STM32Cube with RTOS* (UM1722)
- *Building wireless applications with STM32WB Series microcontrollers* (AN5289)

*Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.*

# 2 STM32CubeWB examples

The examples are classified depending on the STM32Cube level 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, e.g. TIM). Their complexity level ranges from the basic usage of a given peripheral (e.g. PWM generation using timer) to the integration of several peripherals (e.g. how to use DAC for signal generation with synchronization from TIM6 and DMA). 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, e.g. 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, e.g. 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 require high-level firmware bricks (e.g. 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_WB_VX.Y.Z\Projects*\. They all have the same structure:

- *Inc* folder, containing all header files.
- \Src folder, containing the sources 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.txt instructions.

*Note: Refer to "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, LCD display, 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 contains the list of examples provided with STM32CubeWB MCU Package.

*Note:* *STM32CubeMX-generated examples are highlighted with the* MX *STM32CubeMX icon. TrustZone indicates that the example is Arm® TrustZone® enabled.*

Reference materials available on www.st.com/stm32cubefw.

**Table 1. STM32CubeWB firmware examples**

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| **Templates** | - | Starter project | This projects provides a reference template that can be used to build any firmware application. | MX | - | MX | MX | New |
| | **Total number of templates: 4** | | | **1** | **0** | **1** | **1** | 1 |
| **Templates_LL** | - | Starter project | This projects provides a reference template through the LL API that can be used to build any firmware application. | MX | - | MX | MX | New |
| | **Total number of templates_ll: 4** | | | **1** | **0** | **1** | **1** | 1 |
| **Examples** | - | BSP | How to use the different BSP drivers of the external devices mounted on STM32WB5MM-DK board. | X | X | - | - | - |
| | ADC | ADC_AnalogWatchdog | How to use the ADC peripheral to perform conversions with an analog watchdog and out-of-window interrupts enabled. | - | - | MX | - | - |
| | | ADC_MultiChannelSingleConversion | How to use the ADC to convert several channels using the sequencer in Discontinuous mode. Converted data are indefinitely transferred by DMA into an array (circular mode). | - | - | MX | MX | - |
| | | ADC_Oversampling | How to use the ADC to convert a single channel but using the oversampling feature to increase resolution. | - | - | MX | - | - |
| | | ADC_SingleConversion_TriggerSW_IT | How to use the ADC to convert a single channel at each software start. This example uses the interrupt programming model. | - | - | MX | MX | - |
| | | ADC_SingleConversion_TriggerTimer_DMA | How to use the ADC to convert a single channel at each trigger event from a timer. Converted data are indefinitely transferred by DMA into an array (circular mode). | - | - | MX | MX | - |
| | BSP | BSP_Example | This example describes how to use the BSP API. | - | - | MX | MX | New |
| | COMP | COMP_CompareGpioVsVrefInt_IT | How to configure the COMP peripheral to compare the external voltage applied on a specific pin with the internal voltage reference. | - | - | MX | MX | - |
| | | COMP_CompareGpioVsVrefInt_Window_IT | How to make an analog watchdog using the COMP peripheral in Window mode. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples | CRC | 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 | 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 | - | - |
| | CRYP | CRYP_AESModes | How to use the CRYP peripheral to encrypt and decrypt data using AES in chaining modes (ECB, CBC, CTR). | - | - | MX | - | - |
| | | CRYP_DMA | How to use the AES1 peripheral to encrypt and decrypt data using AES 128 algorithm with ECB chaining mode in DMA mode. | - | - | MX | MX | - |
| | CORTEX | CORTEXM_MPU | This example presents the MPU feature. 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 | - | - |
| | | CORTEXM_SysTick | How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs. | - | - | MX | - | - |
| | DMA | DMA_FLASHToRAM | How to use a DMA to transfer a word data buffer from flash memory to embedded SRAM through the HAL API. | - | - | MX | MX | - |
| | | DMA_MUXSYNC | How to use the DMA with the DMAMUX to synchronize a transfer with the LPTIM1 output signal. USART1 is used in DMA synchronized mode to send a countdown from 10 to 00, with a period of 2 seconds. | - | - | MX | MX | - |
| | | DMA_MUX_RequestGen | How to use the DMA with the DMAMUX request generator to generate DMA transfer requests upon EXTI4 signal rising edge. | - | - | MX | MX | - |
| | FLASH | FLASH_EraseProgram | How to configure and use the FLASH HAL API to erase and program the internal flash memory. | - | - | MX | MX | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples | FLASH | FLASH_WriteProtection | How to configure and use the FLASH HAL API to enable and disable the write protection of the internal flash memory. | - | - | MX | MX | - |
| | GPIO | GPIO_EXTI | How to configure external interrupt lines. | - | - | MX | MX | - |
| | | GPIO_IOToggle | How to configure and use GPIOs through the HAL API. | - | - | MX | MX | - |
| | HAL | HAL_TimeBase | How to customize the HAL using a general-purpose timer as main source of timebase, instead of the SysTick. | - | - | MX | MX | - |
| | | HAL_TimeBase_RTC_ALARM | How to customize the HAL using RTC alarm as main source of timebase, instead of the SysTick. | - | - | MX | - | - |
| | | HAL_TimeBase_RTC_WKUP | How to customize the HAL using RTC wakeup as main source of timebase, instead of the SysTick. | - | - | MX | MX | - |
| | | HAL_TimeBase_TIM | How to customize the HAL using a general-purpose timer as main source of timebase instead of the SysTick. | - | - | MX | MX | - |
| | HSEM | HSEM_ProcessSync | How to use a hardware semaphore to synchronize two processes. | - | - | MX | - | - |
| | | HSEM_ReadLock | How to enable, take, then release a semaphore using two different processes. | - | - | MX | - | - |
| | I2C | I2C_TwoBoards_AdvComIT | How to handle multiple I2C data buffer transmissions/receptions between a master and a slave device, using an interrupt. | - | - | MX | - | - |
| | | I2C_TwoBoards_ComDMA | How to handle I2C data buffer transmission/ reception between two boards, via DMA. | - | - | MX | MX | - |
| | | I2C_TwoBoards_ComIT | How to handle I2C data buffer transmission/ reception between two boards, using an interrupt. | - | - | MX | MX | - |
| | | I2C_TwoBoards_ComPolling | How to handle I2C data buffer transmission/ reception between two boards in Polling mode. | - | - | MX | - | - |
| | | I2C_TwoBoards_RestartAdvComIT | How to perform multiple I2C data buffer transmissions/receptions between two boards in Interrupt mode and with restart condition. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK(1) | P-NUCLEO-WB55 .USBDongle(1) | P-NUCLEO-WB55 .Nucleo68(1) | NUCLEO-WB15CC(1) | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples | I2C | I2C_TwoBoards_RestartComIT | How to handle single I2C data buffer transmission/reception between two boards in Interrupt mode and with restart condition. | - | - | MX | - | - |
| | | I2C_WakeUpFromStop | How to handle I2C data buffer transmission/ reception between two boards using an interrupt when the device is in Stop mode. | - | - | MX | - | - |
| | | I2C_WakeUpFromStop2 | How to handle I2C data buffer transmission/ reception between two boards using an interrupt when the device is in Stop 2 mode. | - | - | MX | - | - |
| | IWDG | IWDG_Reset | How to handle the IWDG reload counter and simulate a software fault that generates an MCU 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 lap of time. | - | - | MX | MX | - |
| | LCD | LCD_SegmentsDrive | How to drive an LCD glass using the STM32WBxx HAL driver. | - | - | MX | - | - |
| | LPTIM | LPTIM_PWMExternalClock | How to configure and use the LPTIM to generate a PWM signal at the lowest power consumption, using an external counter clock, through the HAL LPTIM API. | - | - | MX | MX | - |
| | | LPTIM_PWM_LSE | How to configure and use the LPTIM peripheral to generate a PWM signal in low-power mode using the LSE as counter clock, through the HAL LPTIM API. | - | - | MX | - | - |
| | | LPTIM_PulseCounter | How to configure and use the LPTIM peripheral to count pulses, through the LPTIM HAL API. | - | - | MX | MX | - |
| | | LPTIM_Timeout | How to use the LPTIMER peripheral to implement a timeout and wake up the system from a low-power mode, through the HAL LPTIM API, | - | - | MX | - | - |
| | PKA | PKA_ECCscalarMultiplication | How to use the PKA peripheral to execute ECC scalar multiplication. This enables the generation of a public key from a private key. | - | - | MX | - | - |
| | | PKA_ECCscalarMultiplication_IT | How to use the PKA peripheral to execute ECC scalar multiplication. This enables the generation of a public key from a private key in interrupt mode. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples | PKA | PKA_ECDSA_Sign | How to compute a signed message regarding the elliptic curve digital signature algorithm (ECDSA). | - | - | MX | MX | - |
| | | PKA_ECDSA_Sign_IT | How to compute a signed message regarding the elliptic curve digital signature algorithm (ECDSA) in interrupt mode. | - | - | MX | - | - |
| | | PKA_ECDSA_Verify | How to determine if a given signature is valid regarding the elliptic curve digital signature algorithm (ECDSA). | - | - | MX | - | - |
| | | PKA_ECDSA_Verify_IT | How to determine if a given signature is valid regarding the elliptic curve digital signature algorithm (ECDSA) in interrupt mode. | - | - | MX | MX | - |
| | | PKA_ModularExponentiation | How to use the PKA peripheral to execute modular exponentiation. This enables text ciphering/deciphering. | - | - | MX | - | - |
| | | PKA_ModularExponentiationCRT | How to compute the chinese remainder theorem (CRT) optimization. | - | - | MX | - | - |
| | | PKA_ModularExponentiationCRT_IT | How to compute the chinese remainder theorem (CRT) optimization in Interrupt mode. | - | - | MX | - | - |
| | | PKA_ModularExponentiation_IT | How to use the PKA peripheral to execute modular exponentiation. This enables text ciphering/deciphering in Interrupt mode. | - | - | MX | - | - |
| | | PKA_PointCheck | How to use the PKA peripheral to determine if a point is on a curve. This enables the validation of an external public key. | - | - | MX | - | - |
| | | PKA_PointCheck_IT | How to use the PKA peripheral to determine if a point is on a curve. This enables the validation of an external public key. | - | - | MX | - | - |
| | PWR | PWR_LPRUN | How to enter and exit Low-power run mode. | - | - | MX | MX | - |
| | | PWR_LPSLEEP | How to enter Low-power sleep mode and wake up from this mode by using an interrupt. | - | - | MX | MX | - |
| | | PWR_PVD | How to configure the programmable voltage detector by using an external interrupt line. External DC supply must be used to supply $V_{DD}$. | - | - | MX | MX | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples | PWR | PWR_STANDBY_RTC | How to enter Standby mode and wake up from this mode by using an external reset or the RTC wakeup timer. | - | - | MX | MX | - |
| | | PWR_STOP2_RTC | How to enter Stop 2 mode and wake up from this mode using an external reset or RTC wakeup timer. | - | - | MX | - | - |
| | QSPI | QSPI_ExecuteInPlace | How to execute a part of the code from the Quad-SPI memory. To do this, a section is created, where the function is stored. | MX | - | MX | - | - |
| | | QSPI_MemoryMapped | How to erase part of the Quad-SPI memory, write data in DMA mode and access the Quad-SPI memory in Memory-mapped mode to check the data in a forever loop. | MX | - | MX | - | - |
| | | QSPI_ReadWrite_DMA | How to erase part of the Quad-SPI memory, write data in DMA mode, read data in DMA mode and compare the result in a forever loop. | MX | - | MX | - | - |
| | | QSPI_ReadWrite_IT | How to erase part of the Quad-SPI memory, write data in Interrupt mode, read data in Interrupt mode and compare the result in a forever loop. | MX | - | MX | - | - |
| | RCC | RCC_CRS_Synchronization_IT | How to configure the clock recovery service (CRS) in Interrupt mode, using the RCC HAL API. | - | - | MX | - | - |
| | | RCC_CRS_Synchronization_Polling | How to configure the clock recovery service (CRS) in Polling mode, using the RCC HAL API. | - | - | MX | - | - |
| | | RCC_ClockConfig | The main purpose of this example is to serve as a reference for clock configuration operation needed by most of the BLE applications. | MX | - | MX | MX | - |
| | RNG | RNG_MultiRNG | How to configure the RNG using the HAL API. This example uses the RNG to generate 32-bit long random numbers. | - | - | MX | MX | - |
| | | RNG_MultiRNG_IT | How to configure the RNG using the HAL API. This example uses RNG interrupts to generate 32-bit long random numbers. | - | - | MX | - | - |
| | RTC | RTC_Alarm | How to configure and generate an RTC alarm using the RTC HAL API. | - | - | MX | MX | - |
| | | RTC_Calendar | How to configure the calendar using the RTC HAL API. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples | RTC | RTC_LSI | How to use the LSI clock source auto-calibration to get a precise RTC clock. | - | - | MX | - | - |
| | | RTC_Tamper | How to configure the RTC HAL API to write/read data to/from RTC backup registers. | - | - | MX | MX | - |
| | | RTC_TimeStamp | How to configure the RTC HAL API to demonstrate the timestamp feature. | - | - | MX | - | - |
| | SAI | SAI_AudioPlay | How to use the SAI HAL API to play an audio file in DMA circular mode and handle the buffer update. Refer to Projects\STM32WB5MM-DK\Examples\BSP. | X | - | - | - | - |
| | SPI | SPI_FullDuplex_ComDMA_Master | Data buffer transmission/reception between two boards via SPI using DMA. | - | - | MX | MX | - |
| | | SPI_FullDuplex_ComDMA_Slave | Data buffer transmission/reception between two boards via SPI using DMA. | - | - | MX | MX | - |
| | | SPI_FullDuplex_ComIT_Master | Data buffer transmission/reception between two boards via SPI in Interrupt mode. | - | - | MX | - | - |
| | | SPI_FullDuplex_ComIT_Slave | Data buffer transmission/reception between two boards via SPI in Interrupt mode. | - | - | MX | - | - |
| | | SPI_FullDuplex_ComPolling_Master | Data buffer transmission/reception between two boards via SPI in Polling mode. | - | - | MX | - | - |
| | | SPI_FullDuplex_ComPolling_Slave | Data buffer transmission/reception between two boards via SPI in Polling mode. | - | - | MX | - | - |
| | TIM | TIM_DMA | How to use the DMA with TIMER Update request to transfer data from memory to the timer capture compare register 3 (TIMx_CCR3). | - | - | MX | - | - |
| | | TIM_DMABurst | How to update the TIMER channel 1 period and duty cycle using the TIMER DMA burst feature. | - | - | MX | - | - |
| | | TIM_InputCapture | How to use the TIMER peripheral to measure an external signal frequency. | - | - | MX | - | - |
| | | TIM_OCActive | How to configure the TIMER peripheral in Output compare active mode (when the counter matches the capture/compare register, the corresponding output pin is set to its active state). | - | - | MX | MX | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples | TIM | TIM_OCInactive | How to configure the TIMER peripheral in Output compare inactive mode with the corresponding Interrupt requests for each channel. | - | - | MX | - | - |
| | | TIM_OCToggle | How to configure the TIMER peripheral to generate four different signals at four different frequencies. | - | - | MX | - | - |
| | | TIM_OnePulse | How to use of the TIMER peripheral to generate a single pulse when an external signal rising edge is received on the timer input pin. | - | - | X | - | - |
| | | TIM_PWMInput | How to use the TIMER peripheral to measure the frequency and duty cycle of an external signal. | - | - | MX | MX | - |
| | | TIM_PWMOutput | How to configure the TIMER peripheral in PWM (pulse width modulation) mode. | - | - | MX | MX | - |
| | | TIM_TimeBase | How to configure the TIMER peripheral to generate a timebase of one second with the corresponding interrupt request. | - | - | MX | - | - |
| | TSC | TSC_BasicAcquisition_Interrupt | How to use the TSC HAL API to perform continuous acquisitions of one channel in Interrupt mode. | - | - | X | - | - |
| | UART | UART_Console | UART transmission (printf/getchar) via console with user interaction. | MX | - | MX | - | - |
| | | UART_HyperTerminal_DMA | UART transmission (transmit/receive) in DMA mode between a board and an HyperTerminal PC application. | - | - | MX | MX | - |
| | | UART_HyperTerminal_IT | UART transmission (transmit/receive) in Interrupt mode between a board and an HyperTerminal PC application. | - | - | MX | MX | - |
| | | UART_Printf | Re-routing of the C library printf function to the UART. | - | - | MX | MX | - |
| | | UART_ReceptionToIdle_CircularDMA | How to use the HAL UART API for reception to IDLE event in circular DMA mode. | - | - | MX | - | - |
| | | UART_TwoBoards_ComDMA | UART transmission (transmit/receive) in DMA mode between two boards. | - | - | MX | - | - |
| | | UART_TwoBoards_ComIT | UART transmission (transmit/receive) in Interrupt mode between two boards. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| **Examples** | UART | UART_TwoBoards_ComPolling | UART transmission (transmit/receive) in Polling mode between two boards. | - | - | MX | - | - |
| | WWDG | WWDG_Example | How to configure the HAL API to periodically update the WWDG counter and simulate a software fault that generates an MCU WWDG reset when a predefined time period has elapsed. | - | - | MX | MX | - |
| | **Total number of examples: 152** | | | **8** | **1** | **100** | **42** | **1** |
| **Examples_LL** | ADC | ADC_AnalogWatchdog_Init | How to use an ADC peripheral with an analog watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds. | - | - | MX | - | - |
| | | ADC_ContinuousConversion_TriggerSW | How to use an ADC peripheral to perform continuous conversions on a channel, from a software start. | - | - | X | - | - |
| | | ADC_ContinuousConversion_TriggerSW_Init | How to use an ADC peripheral to perform continuous conversions on a channel, from a software start. | - | - | MX | - | - |
| | | ADC_ContinuousConversion_TriggerSW_LowPower_Init | How to use an ADC peripheral with low-power features. | - | - | MX | - | - |
| | | ADC_GroupsRegularInjected_Init | How to use an ADC peripheral with both groups (regular and injected) in their intended use cases. | - | - | MX | - | - |
| | | ADC_MultiChannelSingleConversion_Init | How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence | - | - | - | MX | - |
| | | ADC_Oversampling_Init | How to use an ADC peripheral with oversampling. | - | - | MX | - | - |
| | | ADC_SingleConversion_TriggerSW_DMA_Init | How to use an ADC peripheral to perform a single conversion on a channel, at each software start. This example uses the DMA programming model (for polling or interrupt programming models, refer to other examples). | - | - | MX | - | - |
| | | ADC_SingleConversion_TriggerSW_IT_Init | How to use an ADC peripheral to perform a single conversion on a channel, at each software start. This example uses the interrupt programming model (for polling or DMA programming models, please refer to other examples). | - | - | MX | MX | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_LL | ADC | ADC_SingleConversion_TriggerSW_Init | How to use an ADC peripheral to perform a single conversion on a channel at each software start. This example uses the polling programming model (for interrupt or DMA programming models, please refer to other examples). | - | - | MX | - | - |
| | | ADC_SingleConversion_TriggerTimer_DMA_Init | How to use an ADC peripheral to perform a single conversion on a channel at each trigger event from a timer. Converted data are indefinitely transferred by DMA into a table (circular mode). | - | - | MX | MX | - |
| | | ADC_TemperatureSensor | How to use an ADC peripheral to perform a single conversion on the internal temperature sensor and calculate the temperature in degrees Celsius. | - | - | X | - | - |
| | COMP | COMP_CompareGpioVsVrefInt_IT | How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the internal voltage reference ($V_{REFINT}$), in Interrupt mode. This example is based on the STM32WBxx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |
| | | COMP_CompareGpioVsVrefInt_IT_Init | How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the internal voltage reference ($V_{REFINT}$), in Interrupt mode. This example is based on the STM32WBxx COMP LL API. The peripheral initialization uses the LL initialization function to demonstrate LL initialization usage. | - | - | MX | - | - |
| | | COMP_CompareGpioVsVrefInt_OutputGpio_Init | How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the internal voltage reference ($V_{REFINT}$). The comparator output is connected to a GPIO. This example is based on the STM32WBxx COMP LL API. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_ LL | COMP | COMP_CompareGpioVsVrefInt_Window_IT_Init | How to use a pair of comparator peripherals to compare a voltage level applied on a GPIO pin to two thresholds: the internal voltage reference ($V_{REFINT}$) and a fraction of the internal voltage reference ($V_{REFINT}/2$), in interrupt mode. This example is based on the STM32WBxx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | CORTEX | CORTEX_MPU | This example presents the MPU feature. It configures a memory area as privileged read-only, and attempts to perform read and write operations in different modes. | - | - | MX | - | - |
| | CRC | CRC_CalculateAndCheck | How to configure the CRC calculation unit to compute a CRC code for a given data buffer, based on a fixed generator polynomial (default value 0x4C11DB7). The peripheral initialization is done using LL unitary service functions for optimization purposes (performance and size). | - | - | MX | MX | - |
| | | CRC_UserDefinedPolynomial | How to configure and use the CRC calculation unit to compute an 8-bit CRC code for a given data buffer, based on a user-defined generating polynomial. The peripheral initialization is done using LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | CRS | CRS_Synchronization_IT | How to configure the clock recovery service in Interrupt mode through the STM32WBxx CRS LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | CRS_Synchronization_Polling | How to configure the clock recovery service in polling mode through the STM32WBxx CRS LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | DMA | DMA_CopyFromFlashToMemory | How to use a DMA channel to transfer a word data buffer from flash memory to embedded SRAM. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_ LL | DMA | DMA_CopyFromFlashToMemory_Init | How to use a DMA channel to transfer a word data buffer from flash memory to embedded SRAM. The peripheral initialization uses LL initialization functions to demonstrate LL initialization usage. | - | - | MX | MX | - |
| | EXTI | EXTI_ToggleLedOnIT | How to configure the EXTI and use GPIOs to toggle the user LEDs available on the board when a user button is pressed. It is based on the STM32WBxx LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |
| | | EXTI_ToggleLedOnIT_Init | How to configure the EXTI and use GPIOs to toggle the user LEDs available on the board when a user button is pressed. This example is based on the STM32WBxx LL API. The peripheral initialization is done using LL initialization function to demonstrate LL initialization usage. | - | - | MX | MX | - |
| | GPIO | GPIO_InfiniteLedToggling | How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32WBxx LL API. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | X | - | - |
| | | 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 STM32WBxx LL API. The peripheral is initialized with LL initialization function to demonstrate LL initialization usage. | - | - | MX | MX | - |
| | HSEM | HSEM_DualProcess | How to use the low-layer HSEM API to initialize, lock, and unlock hardware semaphore in the context of two processes accessing the same resource. | - | - | MX | - | - |
| | | HSEM_DualProcess_IT | How to use the low-layer HSEM API to initialize, lock, and unlock hardware semaphore in the context of two processes accessing the same resource. | - | - | MX | MX | - |
| | I2C | I2C_OneBoard_AdvCommunication_DMAAndIT_Init | How to exchange data between an I2C master device in DMA mode and an I2C slave device in Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_LL | I2C | I2C_OneBoard_Communication_DMAAndIT_Init | How to transmit data bytes from an I2C master device using DMA mode to an I2C slave device using Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | - | - |
| | | I2C_OneBoard_Communication_IT | How to handle the reception of one data byte from an I2C slave device by an I2C master device. Both devices operate in Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | X | - | - |
| | | I2C_OneBoard_Communication_IT_Init | How to handle the reception of one data byte from an I2C slave device by an I2C master device. Both devices operate in Interrupt mode. The peripheral is initialized with LL initialization function to demonstrate LL initialization usage. | - | - | MX | - | - |
| | | I2C_OneBoard_Communication_PollingAndIT_Init | How to transmit data bytes from an I2C master device using Polling mode to an I2C slave device using Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | - | - |
| | | I2C_TwoBoards_MasterRx_SlaveTx_IT_Init | How to handle the reception of one data byte from an I2C slave device by an I2C master device. Both devices operate in Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | MX | - |
| | | I2C_TwoBoards_MasterTx_SlaveRx_DMA_Init | How to transmit data bytes from an I2C master device using DMA mode to an I2C slave device using DMA mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | MX | - |
| | | I2C_TwoBoards_MasterTx_SlaveRx_Init | How to transmit data bytes from an I2C master device using Polling mode to an I2C slave device in Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | - | - |
| | | I2C_TwoBoards_WakeUpFromStop2_IT_Init | How to handle the reception of a data byte from an I2C slave device in Stop 2 mode by an I2C master device. Both devices operate in Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_LL | I2C | I2C_TwoBoards_WakeUpFromStop_IT_Init | How to handle the reception of a data byte from an I2C slave device in Stop 1 mode by an I2C master device. Both devices operate in Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | 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 (SW1) is pressed. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | MX | MX | - |
| | LPTIM | LPTIM_PulseCounter | How to use the LPTIM peripheral in Counter mode to generate a PWM output signal and update its duty cycle. This example is based on the STM32WBxx LPTIM LL API. The peripheral is initialized with LL unitary service functions to optimize performance and size. | - | - | X | - | - |
| | | LPTIM_PulseCounter_Init | How to use the LPTIM peripheral in Counter mode to generate a PWM output signal and update its duty cycle. This example is based on the STM32WBxx LPTIM LL API. The peripheral is initialized with LL initialization function to demonstrate LL initialization usage. | - | - | MX | MX | - |
| | LPUART | LPUART_WakeUpFromStop2_Init | How to configure GPIO and LPUART peripherals to allow characters received on 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 LL initialization function to demonstrate LL initialization usage. | - | - | MX | - | - |
| | | LPUART_WakeUpFromStop_Init | How to configure GPIO and LPUART peripherals to allow characters received on 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 LL initialization function to demonstrate LL initialization usage. | - | - | MX | MX | - |
| | PKA | PKA_ECDSA_Sign | How to use the low-layer PKA API to generate an ECDSA signature. | - | - | MX | MX | - |
| | | PKA_ModularExponentiation | How to use the low-layer PKA API to execute RSA modular exponentiation. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_ LL | PWR | PWR_EnterStandbyMode | How to enter Standby mode and wake up from this mode by using an external reset or a wakeup interrupt. | - | - | MX | MX | - |
| | | PWR_EnterStopMode | How to enter Stop 2 mode. | - | - | MX | MX | - |
| | | PWR_OptimizedRunMode | How to increase/decrease frequency and $V_{CORE}$ and enter/exit Low-power run mode. | - | - | MX | MX | - |
| | | PWR_SMPS_16MHZ_HSI | How to use STM32WBxx power converters (SMPS, LDO and LP-LDO) depending on $V_{DD}$ voltage and low-power mode. | - | - | MX | - | - |
| | | PWR_SMPS_64MHZ_MSI_PLL | How to use STM32WBxx power converters (SMPS, LDO and LP-LDO) depending on $V_{DD}$ voltage and low-power mode. | - | - | MX | - | - |
| | RCC | RCC_HWAutoMSICalibration | How to use of the MSI clock source hardware auto-calibration and LSE clock (PLL mode) to obtain an accurate MSI clock. | - | - | MX | - | - |
| | | RCC_OutputSystemClockOnMCO | How to configure the MCO pin (PA8) to output the system clock. | - | - | MX | - | - |
| | | RCC_UseHSEasSystemClock | How to use the RCC LL API to start the HSE and use it as system clock. | - | - | MX | - | - |
| | | RCC_UseHSI_PLLasSystemClock | How to modify the PLL parameters in run time. | - | - | MX | MX | - |
| | RNG | RNG_GenerateRandomNumbers | How to configure the RNG to generate 32-bit long random numbers. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | RNG_GenerateRandomNumbers_IT | How to configure the RNG to generate 32-bit long random numbers using interrupts. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | MX | - |
| | RTC | RTC_Alarm | How to configure the RTC LL API to configure and generate an alarm using the RTC peripheral. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_ LL | RTC | RTC_Alarm_Init | How to configure the RTC LL API to configure and generate an alarm using the RTC peripheral. The peripheral initialization uses the LL initialization function. | - | - | MX | - | - |
| | | RTC_Calendar_Init | How to configure the LL API to set the RTC calendar. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | RTC_ExitStandbyWithWakeUpTimer_Init | How to configure the RTC to wake up from Standby mode using the RTC wakeup timer. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | RTC_Tamper_Init | How to configure the Tamper using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | RTC_TimeStamp_Init | How to configure the Timestamp using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | SPI | SPI_OneBoard_HalfDuplex_DMA | How to configure GPIO and SPI peripherals to transmit bytes from an SPI Master device to an SPI Slave device in DMA mode. This example is based on the STM32WBxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |
| | | SPI_OneBoard_HalfDuplex_DMA_Init | How to configure GPIO and SPI peripherals to transmit bytes from an SPI Master device to an SPI Slave device in DMA mode. This example is based on the STM32WBxx SPI LL API. The peripheral initialization uses the LL initialization function to demonstrate LL initialization usage. | - | - | MX | - | - |
| | | SPI_OneBoard_HalfDuplex_IT_Init | How to configure GPIO and SPI peripherals to transmit bytes from an SPI Master device to an SPI Slave device in Interrupt mode. This example is based on the STM32WBxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_ LL | SPI | SPI_TwoBoards_FullDuplex_ DMA_Master_Init | Data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32WBxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | MX | - |
| | | SPI_TwoBoards_FullDuplex_ DMA_Slave_Init | Data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32WBxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | MX | - |
| | | SPI_TwoBoards_FullDuplex_I T_Master_Init | Data buffer transmission and reception via SPI using Interrupt mode. This example is based on the STM32WBxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | SPI_TwoBoards_FullDuplex_I T_Slave_Init | Data buffer transmission and reception via SPI using Interrupt mode. This example is based on the STM32WBxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | TIM | TIM_BreakAndDeadtime | How to configure the TIMER peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined dead time value, use the break feature, and lock the break and dead-time configuration. | - | - | X | - | - |
| | | TIM_DMA_Init | How to use the DMA with a timer update request to transfer data from memory to timer capture compare register 3 (TIMx_CCR3). This example is based on the STM32WBxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | TIM_InputCapture_Init | How to use the TIMER peripheral to measure a periodic signal frequency provided either by an external signal generator or by another timer instance. This example is based on the STM32WBxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_ LL | TIM | TIM_OnePulse | How to configure the TIMER to generate a positive pulse in Output compare mode with a length of $t_{PULSE}$ and after a delay of $t_{DELAY}$. This example is based on the STM32WBxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |
| | | TIM_OutputCompare_Init | How to configure the TIMER peripheral to generate an output waveform in different Output compare modes. This example is based on the STM32WBxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | | TIM_PWMOutput | How to use of the TIMER peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32WBxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |
| | | TIM_PWMOutput_Init | How to use the TIMER peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32WBxx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL initialization. | - | - | MX | - | - |
| | | TIM_TimeBase_Init | How to configure the TIMER peripheral to generate a timebase. This example is based on the STM32WBxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | MX | - | - |
| | USART | USART_Communication_Rx_IT | How to configure GPIO and USART peripherals to receive characters from an HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size). | - | - | X | - | - |
| | | USART_Communication_Rx_IT_Continuous_Init | How to configure GPIO and USART peripheral to continuously receive characters from HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size). | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_LL | USART | USART_Communication_Rx_IT_Continuous_VCP_Init | How to configure GPIO and USART peripheral to continuously receive characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size). | - | - | MX | - | - |
| | | USART_Communication_Rx_IT_Init | How to configure GPIO and USART peripheral to receive characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. The peripheral initialization is done using LL initialization function to demonstrate LL initialization usage. | - | - | MX | MX | - |
| | | USART_Communication_Rx_IT_VCP_Init | How to configure GPIO and USART peripheral to receive characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. The peripheral initialization is done using LL initialization function to demonstrate LL initialization usage. | - | - | MX | - | - |
| | | USART_Communication_TxRx_DMA_Init | How to configure GPIO and USART peripheral to send characters asynchronously to/from an HyperTerminal (PC) in DMA mode. This example is based on STM32WBxx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size). | - | - | MX | - | - |
| | | USART_Communication_Tx_IT_Init | How to configure GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on STM32WBxx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size). | - | - | MX | - | - |
| | | USART_Communication_Tx_IT_VCP_Init | How to configure GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on STM32WBxx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size). | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_LL | USART | USART_Communication_Tx_Init | How to configure GPIO and USART peripherals to send characters asynchronously to an HyperTerminal (PC) in Polling mode. If the transfer cannot be completed within the allocated time, a timeout enables exiting from the sequence with a Timeout error code. This example is based on STM32WBxx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size). | - | - | MX | MX | - |
| | | USART_Communication_Tx_VCP_Init | How to configure GPIO and USART peripherals to send characters asynchronously to an HyperTerminal (PC) in Polling mode. If the transfer cannot be completed within the allocated time, a timeout enables exiting from the sequence with a Timeout error code. This example is based on STM32WBxx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size). | - | - | MX | - | - |
| | | USART_WakeUpFromStop1_Init | How to configure GPIO and USART1 peripherals to enable the characters received on USART_RX pin to wake up the MCU from low-power mode. | - | - | MX | - | - |
| | | USART_WakeUpFromStop_Init | How to configure GPIO and USART1 peripherals to enable the characters received on USART_RX pin to wake up the MCU from low-power mode. | - | - | MX | - | - |
| | UTILS | UTILS_ConfigureSystemClock | How to use UTILS LL API to configure the system clock using PLL with HSI as source clock. | - | - | MX | - | - |
| | | UTILS_ReadDeviceInfo | How to read the UID, Device ID and Revision ID and save them into a global information buffer. | - | - | MX | MX | - |
| | WWDG | WWDG_RefreshUntilUserEvent_Init | How to configure the WWDG to periodically update the counter and generate an MCU WWDG reset when a user button is pressed. The peripheral initialization uses the LL unitary service functions for optimization purposes (performance and size). | - | - | MX | MX | - |
| | **Total number of examples_ll: 118** | | | **0** | **0** | **92** | **26** | **0** |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Examples_ MIX | ADC | ADC_SingleConversion_TriggerSW_IT | How to use the ADC to perform a single ADC channel conversion at each software start. This example uses the interrupt programming model (for polling and DMA programming models, please refer to other examples). It is based on the STM32WBxx ADC HAL and LL API. The LL API is used for performance improvement. | - | - | MX | - | - |
| | CRC | CRC_PolynomialUpdate | How to use the CRC peripheral through the STM32WBxx CRC HAL and LL API. | - | - | MX | - | - |
| | DMA | DMA_FLASHToRAM | How to use a DMA to transfer a word data buffer from flash memory to embedded SRAM through the STM32WBxx DMA HAL and LL API. The LL API is used for performance improvement. | - | - | MX | MX | - |
| | I2C | I2C_OneBoard_ComSlave7_10bits_IT | How to perform I2C data buffer transmission/ reception between one master and two slaves with different address sizes (7-bit or 10-bit). This example uses the STM32WBxx I2C HAL and LL API (LL API usage for performance improvement) and an interrupt. | - | - | MX | - | - |
| | PWR | PWR_STOP1 | How to enter Stop 1 mode and wake up from this mode by using external reset or wakeup interrupt (all the RCC function calls use RCC LL API for minimizing footprint and maximizing performance). | - | - | MX | MX | - |
| | SPI | SPI_FullDuplex_ComPolling_Master | Data buffer transmission/reception between two boards via SPI in Polling mode. | - | - | MX | - | - |
| | | SPI_FullDuplex_ComPolling_Slave | Data buffer transmission/reception between two boards via SPI in Polling mode. | - | - | MX | - | - |
| | | SPI_HalfDuplex_ComPollingIT_Master | Data buffer transmission/reception between two boards via SPI in Polling (LL driver) and Interrupt modes (HAL driver). | - | - | MX | MX | - |
| | | SPI_HalfDuplex_ComPollingIT_Slave | Data buffer transmission/reception between two boards via SPI in Polling mode (LL driver) and Interrupt modes (HAL driver). | - | - | MX | MX | - |
| | TIM | TIM_PWMInput | How to use the TIMER peripheral to measure an external signal frequency and duty cycle. | - | - | MX | MX | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| **Examples_ MIX** | UART | UART_HyperTerminal_IT | How to use a UART to transmit/receive data between a board and an HyperTerminal PC application in Interrupt mode. This example describes how to use the USART peripheral through the STM32WBxx UART HAL and LL API, the LL API being used for performance improvement. | - | - | MX | MX | - |
| | | UART_HyperTerminal_TxPolling_RxIT | How to use a UART to transmit/receive data between a board and an HyperTerminal PC application both in Polling and Interrupt modes. This example describes how to use the USART peripheral through the STM32WBxx UART HAL and LL API, the LL API being used for performance improvement. | - | - | MX | - | - |
| | **Total number of examples_mix: 18** | | | **0** | **0** | **12** | **6** | |
| **Applications** | BLE | BLE_AT_Server | How to demonstrate Point-to-Point communication using BLE component (as GATT server). | - | - | X | - | - |
| | | BLE_Beacon | How to advertize three types of beacon (tlm, uuid, url). | - | - | MX | - | - |
| | | BLE_BloodPressure | How to use the Blood Pressure profile as specified by the BLE SIG. | - | - | MX | - | - |
| | | BLE_CableReplacement | How to use the point-to-point communication using BLE component. | - | - | X | - | - |
| | | BLE_Custom | This example is to demonstrate that a BLE_Custom application can be created using STM32CubeMX. | - | - | MX | - | - |
| | | BLE_DataThroughput | How to use data throughput via notification from server to client using BLE component. | - | - | X | X | - |
| | | BLE_HealthThermometer | How to use the Health Thermometer profile as specified by the BLE SIG. | - | - | MX | - | - |
| | | BLE_HeartRate | How to use the Heart Rate profile as specified by the BLE SIG. | MX | X | MX | MX | - |
| | | BLE_HeartRateFreeRTOS | How to use the Heart Rate profile as specified by the BLE SIG with FreeRTOS. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| **Applications** | BLE | BLE_HeartRateFreeRTOS_ANCS | How to read notifications from Apple® Notification Center Service (ANCS) as specified by Apple specification, and use the Heart Rate profile as specified by the BLE SIG with FreeRTOS. | - | - | X | - | - |
| | | BLE_HeartRateThreadX | How to use the Heart Rate profile as specified by the BLE SIG using ThreadX OS. | - | - | X | - | - |
| | | BLE_HeartRate_ANCS | How to read notifications from Apple Notification Center Service (ANCS) as specified by Apple specification, and use Heart Rate profile as specified by the BLE SIG. | - | - | X | - | - |
| | | BLE_HeartRate_ota | How to use the Heart Rate profile as specified by the BLE SIG to be downloaded with BLE OTA application. | - | - | X | X | - |
| | | BLE_Hid | How to use the Human Interface Device profile as specified by the BLE SIG. | - | - | X | - | - |
| | | BLE_HR_P2PServer | Includes two BLE services: the first one is a BLE_P2P_Server like, including two characteristics, the second one is a BLE_Heart_Rate like with three characteristics. | - | - | - | - | New |
| | | BLE_MeshLightingLPN | This is the implementation of the BLE Mesh Low Power Node profile as specified by the BLE SIG. | - | X | X | - | - |
| | | BLE_MeshLightingPRFNode | This is the implementation of the BLE Mesh Lighting profile as specified by the BLE SIG. | - | X | X | - | - |
| | | BLE_MeshLightingProvisioner | This is the implementation of the BLE Mesh Lighting profile as specified by the BLE SIG. | - | - | X | - | - |
| | | BLE_Mesh_Model_Sensor | This is the implementation of a BLE Mesh Sensor Model (client and server) as specified by the BLE SIG. | X | - | - | - | - |
| | | BLE_Mesh_ThermometerSensor | This is the implementation of a BLE Mesh Vendor profile as specified by the BLE SIG. | X | - | - | - | - |
| | | BLE_MultiAppAt | How to use multiBLE applications using a network processor architecture. | - | - | - | - | - |
| | | BLE_Ota | OTA implementation to download a new image into the user flash memory. | - | - | X | X | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Applications | BLE | BLE_Peripheral_Lite | How to communicate with simple BLE peripheral with the features minimum activated. | - | - | X | - | - |
| | | BLE_Proximity | How to use the Proximity profile as specified by the BLE SIG. | - | - | X | - | - |
| | | BLE_RfWithFlash | How to demonstrate the capability to erase/ write the flash memory while a point-to-point communication using BLE component is active. | - | - | X | - | - |
| | | BLE_Sensor | This example demonstrates the capabilities of STM32WB5MM Discovery kit with the use of board sensors. | MX | - | - | - | - |
| | | BLE_TransparentMode | How to communicate with the STM32CubeMonitor-RF tool using the Transparent mode. | MX | - | MX | MX | - |
| | | BLE_TransparentModeVCP | How to communicate with the STM32CubeMonitor-RF tool using the Transparent mode. | - | X | - | - | - |
| | | BLE_p2pClient | How to demonstrate point-to-point communication using BLE component (as GATT client). | - | X | MX | MX | - |
| | | BLE_p2pClient_Ext | Demontrates a BLE scanner with connections from an extended and a legacy advertising Two Nucleo STM32WB55xx boards (MB1355C) are used, one acting as GATT client, and one as GATT server. | - | - | X | X | - |
| | | BLE_p2pRouteur | This example demonstrates Multipoint communication using BLE component. | - | X | MX | - | - |
| | | BLE_p2pServer | How to demonstrate point-to-point communication using BLE component (as GATT server). | MX | X | MX | MX | - |
| | | BLE_p2pServer_Ext | Demonstrates multiple extended advertising sets. | - | - | X | X | - |
| | | BLE_p2pServer_ota | How to demonstrate point-to-point communication using BLE component (peripheral such as GATT server) to be downloaded with BLE OTA application. | - | - | X | X | - |
| | BLE_LLD | BLE_LLD_Chat | How to create a "Chat" talk between two boards using terminals. | X | - | X | X | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK(1) | P-NUCLEO-WB55 .USBDongle(1) | P-NUCLEO-WB55 .Nucleo68(1) | NUCLEO-WB15CC(1) | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Applications | BLE_LLD | BLE_LLD_Datarate | How to send BLE LLD packets in high data rate. | - | X | X | - | - |
| | | BLE_LLD_Lowpower | How to send BLE LLD packets in low-power mode. | - | - | X | X | - |
| | | BLE_LLD_Pressbutton | How to control remote LEDs with BLE LLD. | - | - | X | X | - |
| | | BLE_LLD_Proximity | How to use BLE LLD to detect nearby boards. This solution can be used in contact tracing for Covid-19, as it provides an estimation of the distance with other boards. | - | - | X | - | - |
| | BLE_Mac | BLE_Mac_Static | How to use BLE and 802_15_4 Mac applications in Static concurrent mode. | - | - | X | - | - |
| | BLE_Thread | BLE_Thread_Dyn | How to use BLE and Thread applications in Dynamic concurrent mode. | - | - | X | - | - |
| | | BLE_Thread_Dyn_SED | How to use BLE and Thread applications (acting as sleep-end device) in Dynamic concurrent mode. | - | - | X | - | - |
| | | BLE_Thread_Static | How to use BLE and Thread applications in Static concurrent mode. | - | - | X | - | - |
| | BLE_Zigbee | BLE_Zigbee_Dyn | How to use BLE and Zigbee® applications (acting as router) in Dynamic concurrent mode. | - | - | X | - | - |
| | | BLE_Zigbee_Dyn_SED | How to use BLE and Zigbee applications (acting as sleep end device) in Dynamic concurrent mode. | - | - | - | - | - |
| | | BLE_Zigbee_Static | How to use BLE and Zigbee applications in Static concurrent mode. | - | - | X | - | - |
| | CKS | CKS_Crypt | How to use the CKS feature to store AES cryptographic keys in secure area. | - | - | X | - | - |
| | Demonstrations | Audio_BVLINKWB | This demonstration firmware is based on STM32Cube Function pack for STM32WB MCUs featuring full-duplex audio streaming over Bluetooth 5.0 using Opus codec. | X | - | - | - | - |
| | FreeRTOS | FreeRTOS_Mail | How to use mail queues with CMSIS RTOS API. | - | - | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Applications | FreeRTOS | FreeRTOS_Mutexes | How to use mutexes with CMSIS RTOS API. | - | - | MX | MX | - |
| | | FreeRTOS_Queues | How to use message queues with CMSIS RTOS API. | - | - | MX | - | - |
| | | FreeRTOS_Semaphore | How to use semaphores with CMSIS RTOS API. | - | - | MX | - | - |
| | | FreeRTOS_SemaphoreFromISR | How to use semaphore from ISR with CMSIS RTOS API. | - | - | MX | - | - |
| | | FreeRTOS_Signal | How to perform thread signaling using CMSIS RTOS API. | - | - | MX | - | - |
| | | FreeRTOS_SignalFromISR | How to use CMSIS-OS Signal API from ISR context. | - | - | MX | - | - |
| | | FreeRTOS_ThreadCreation | How to implement thread creation using CMSIS RTOS API. | - | - | MX | - | - |
| | | FreeRTOS_Timers | How to use timers of CMSIS RTOS API. | - | - | MX | - | - |
| | Mac_802_15_4 | Mac_802_15_4_FFD | How to use MAC 802.15.4 Association and Data exchange. | - | - | X | - | - |
| | | Mac_802_15_4_LPM_Periodic_Transmit | How to use MAC 802.15.4 data transmission with Stop 1 low power mode enabled. | - | - | X | - | - |
| | | Mac_802_15_4_RFD | How to use MAC 802.15.4 Association and Data exchange. | - | - | X | - | - |
| | Phy_802_15_4 | Phy_802_15_4_Cli | How to create a "PHY_802.15.4 command line interface" application on STM32WB55xx boards using terminals. | - | - | X | - | - |
| | Thread | Thread_Cli_Cmd | How to control the Thread stack via Cli commands. | - | X | MX | - | - |
| | | Thread_Coap_DataTransfer | How to transfer large blocks of data through the CoAP messaging protocol. | - | X | MX | - | - |
| | | Thread_Coap_Generic | How to build Thread application based on CoAP messages. | X | X | MX | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| **Applications** | Thread | Thread_Coap_Generic_Ota | How to build Thread application based on CoAP messages. | - | - | X | - | - |
| | | Thread_Coap_Generic_ThreadX | How to build Thread application based on Coap messages. (using ThreadX) This application requires two STM32WB55xx boards. | - | - | X | - | - |
| | | Thread_Coap_MultiBoard | How to use CoAP for sending message to multiple boards. | - | - | MX | - | - |
| | | Thread_Coap_Secure | How to build Thread application based on CoAP Secure messages. | - | - | X | - | - |
| | | Thread_Commissioning | How to use Thread commissioning process. | - | - | MX | - | - |
| | | Thread_FTD_Coap_Multicast | How to exchange multicast CoAP messages. | - | X | MX | - | - |
| | | Thread_NVM | How to configure NVM for Thread applications. | - | - | X | - | - |
| | | Thread_Ota | How to update over-the-air (OTA) firmware application and Copro Wireless binary using Thread. | - | - | X | - | -- |
| | | Thread_Ota_Server | How to update over-the-air (OTA) firmware application and Copro Wireless binary using Thread. | - | - | X | - | - |
| | | Thread_RCP | This application is used to demonstrate the OpenThread Border router feature using an STM32WB device. | - | - | X | - | - |
| | | Thread_RCP_Cli_Cmd | How to control the Thread stack via Cli commands and trigger an automatic commissioning joiner sequence from a press button. | - | - | X | - | - |
| | | Thread_SED_Coap_FreeRTOS | How to exchange a CoAP message using the Thread protocol. | - | - | MX | - | - |
| | | Thread_SED_Coap_Multicast | How to exchange a CoAP message using the Thread protocol. | - | X | MX | - | - |
| | | Thread_Udp | How to transfer data using UDP. | - | - | X | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Applications | TouchSensing | TouchSensing_1touchKey | Use of the STMTouch driver with one touchkey sensor. | - | - | X | - | - |
| | | TouchSensing_1touchkey | Use of the STMTouch driver with one touchkey sensor. | MX | - | - | - | - |
| | USB_Device | CDC_Standalone | How to use USB device application based on the Device Communication Class (CDC) following the PSTN sub-protocol on the STM32WB MCUs. | - | - | MX | - | - |
| | | DFU_Standalone | Compliant implementation of the device firmware upgrade (DFU). | - | MX | MX | - | - |
| | | HID_Standalone | How to use of the USB device application based on the human interface (HID). | - | MX | MX | - | - |
| | Zigbee | Zigbee_APS_Coord | How to use the APS layer in an application with a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_APS_Router | How to use the APS layer in an application with a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_Commissioning_Client_Coord | How to use Commissioning cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_Commissioning_Server_Router | How to use Commissioning cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_DevTemp_Client_Router | How to use DevTemp cluster on a centralized Zigbee network with device acting as router. | - | - | X | - | - |
| | | Zigbee_DevTemp_Server_Coord | How to use DevTemp cluster on a centralized Zigbee network with device acting as server. | - | - | X | - | - |
| | | Zigbee_Diagnostic_Client_Router | How to use Diagnostic cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_Diagnostic_Server_Coord | How to use Diagnostic cluster as a server on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_DoorLock_Client_Router | How to use Door Lock cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_DoorLock_Server_Coord | How to use Door Lock cluster as a client on a centralized Zigbee network. | - | - | X | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| Applications | Zigbee | Zigbee_Find_Bind_Coord | How to use Finding and Binding on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_Find_Bind_IAS_Router2 | How to use Finding and Binding on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_Find_Bind_OnOff_Router1 | How to use Finding and Binding on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_IAS_WD_Client_Router | How to use IAS WD cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_IAS_WD_Server_Coord | How to use IAS WD cluster as a server on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_MeterId_Client_Router | How to use Meter Identification cluster as a client on a centralized Zigbee network. | - | - | MX | - | - |
| | | Zigbee_MeterId_Server_Coord | How to use Meter Identification cluster as a server on a centralized Zigbee network. | - | X | MX | - | - |
| | | Zigbee_OnOff_ChannelsAgility_SED | How to use the OnOff cluster on a device on a Sleepy End Device (SED) acting as a Client within a Centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_ChannelsAgility_SEDRejoin | How to use the OnOff cluster on a device acting as a Client with EndDevice role within a Centralized Zigbee network. | - | - | New | - | - |
| | | Zigbee_OnOff_ChannelsAgility_ZC | How to use the OnOff cluster on a device acting acting as a Server with Coordinator role within a Centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_ChannelsAgility_ZR | How to use the OnOff cluster on a device acting as a Client with Router role within a Centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_ClientRouter_ThreadX | How to use the OnOff cluster on a device acting as a Client with Router role within a Centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_Server_Coord_ThreadX | How to use the OnOff cluster on a device acting as a Server with Coordinator role within a Centralized Zigbee network using ThreadX. | - | - | New | - | - |
| | | Zigbee_OTA_Client_Router | How to use OTA cluster as a client on a centralized Zigbee network. | - | - | X | - | - |

| Level | Module name | Project name | Description | STM32WB 5MM-DK[1] | P-NUCLEO-WB55 .USBDongle[1] | P-NUCLEO-WB55 .Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| **Applications** | Zigbee | Zigbee_OTA_Server_Coord | How to use OTA cluster as a server on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_Client_Distrib | How to use OnOff cluster as a client on a distributed Zigbee network. | - | - | MX | - | - |
| | | Zigbee_OnOff_Client_Router | How to use OnOff cluster as a client on a centralized Zigbee network. | X | X | MX | - | - |
| | | Zigbee_OnOff_Client_Router_FreeRTOS | How to use OnOff cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_Client_Router_Ota | How to use OnOff cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_Client_SED | How to use OnOff cluster as a client on a centralized Zigbee network. | - | - | MX | - | - |
| | | Zigbee_OnOff_Coord_NVM | How to use OnOff cluster with persistent data on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_Router_NVM | How to use OnOff cluster with persistent data on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_Server_Coord | How to use OnOff cluster as a server on a centralized Zigbee network. | X | X | MX | - | - |
| | | Zigbee_OnOff_Server_Coord_FreeRTOS | How to use OnOff cluster as a server on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_OnOff_Server_Distrib | How to use OnOff cluster as a server on a distributed Zigbee network. | - | - | MX | - | - |
| | | Zigbee_PollControl_Client_Coord | How to use Poll Control cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_PollControl_Server_SED | How to use Poll Control cluster as a server on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_PowerProfile_Client_Coord | How to use Power Profile cluster as a client on a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_PowerProfile_Server_Router | How to use Power Profile cluster as a server on a centralized Zigbee network. | - | X | X | - | - |

| Level | Module name | Project name | Description | STM32WB5MM-DK[1] | P-NUCLEO-WB55.USBDongle[1] | P-NUCLEO-WB55.Nucleo68[1] | NUCLEO-WB15CC[1] | B-WB1M-WPAN1 |
|---|---|---|---|---|---|---|---|---|
| **Applications** | Zigbee | Zigbee_PressMeas_Client_Router | How to use Pressure Measurement cluster on a centralized Zigbee network with device acting as router. | - | - | X | - | - |
| | | Zigbee_PressMeas_Server_Coord | How to use Pressure Measurement cluster on a centralized Zigbee network with device acting as server. | - | - | X | - | - |
| | | Zigbee_SE_Msg_Client_Coord | How to use SE Messaging cluster on a centralized Zigbee network with device acting as coordinator (client). | - | - | X | - | - |
| | | Zigbee_SE_Msg_Server_Router | How to use SE Messaging cluster on a centralized Zigbee network with device acting as router (server). | - | - | X | - | - |
| | | Zigbee_TempMeas_Client_Router | How to use TempMeas as a client on a centralized Zigbee network. | X | - | - | - | - |
| | | Zigbee_TempMeas_Server_Coord | How to use Temperature Measurement cluster as a server on a centralized Zigbee network. | X | - | - | - | - |
| | | Zigbee_custom_ls_Client_Router | How to use the Custom long string cluster on a device acting as a Client with Router role within a centralized Zigbee network. | - | - | X | - | - |
| | | Zigbee_custom_ls_Server_Coord | How to use the Custom long string cluster on a device acting as a server with coordinator role within a centralized Zigbee network. | - | - | X | - | - |
| | **Total number of applications: 169** | | | **14** | **19** | **121** | **14** | **1** |
| **Total number of projects: 465** | | | | **24** | **20** | **327** | **90** | **4** |

1. *STM32CubeMX-generated examples are highlighted with the MX STM32CubeMX icon. Other examples are marked with x. They are specific TrustZone examples, if marked as such.*

# Revision history

Table 2. **Document revision history**

| Date | Version | Changes |
|---|---|---|
| 19-Feb-2019 | 1 | Initial release. |
| 17-Mar-2020 | 2 | Added Zigbee middleware in Figure 1. STM32CubeWB firmware components.<br>Added *STM32CubeWB Nucleo demonstration firmware* (UM2251) in Section 1 Reference documents.<br>Added \STM32CubeIDE in the list of project folders in Section 2 STM32CubeWB examples.<br>Updated Table 1. STM32CubeWB firmware examples |
| 09-Nov-2020 | 3 | Updated folder hosting the examples as well of folder structure in Section 2 STM32CubeWB examples.<br>Table 1. STM32CubeWB firmware examples:<br>• Suppressed NUCLEO-WB35CE board<br>• Removed *CORTEXM_SysTick* and *I2S_Audio* examples<br>• Added *BLE_Custom*, *BLE_HeartRateFreeRTOS_ANCS*, *BLE_HeartRate_ANCS*, *BLE_MeshLightingLPN*, *BLE_MeshLightingProvisioner,* , *BLE_RfWithFlash*, *BLE_Thread_Dyn*, *BLE_Thread_Dyn_SED*, *BLE_Zigbee_Dyn*, *BLE_Zigbee_Dyn_NVM*, *BLE_Zigbee_Dyn_SED*, LLD_BLE, Phy_802_15_4, *Thread_Upd*, *Zigbee_APS_Coord*, *Zigbee_APS_Router*, *Zigbee_Commissioning_Client_Coord* and *Zigbee_Commissioning_Client_Router*, *Zigbee_Diagnostic_Client_Router*, *Zigbee_Diagnostic_Server_Coord*, *Zigbee_DoorLock_Client_Router*, *Zigbee_DoorLock_Server_Coord*, *Zigbee_IAS_WD_Client_Router*, *Zigbee_IAS_WD_Server_Coord*, *Zigbee_OTA_Client_Router*, *Zigbee_OTA_Server_Coord*, *Zigbee_OnOff_Client_Router_Ota*, *Zigbee_OnOff_Client_SED*, *Zigbee_OnOff_Coord_NVM*, *Zigbee_OnOff_Router_NVM*, *Zigbee_PollControl_Client_Coord*, *Zigbee_PollControl_Server_SED* applications |
| 26-Apr-2021 | 4 | Updated Table 1. STM32CubeWB firmware examples:<br>• Added STM32WB5MM-DK kit and associated examples.<br>• Added CORTEX CORTEXM_SysTick, UART_Console and UART_ReceptionToIdle_CircularDMA examples.<br>• Updated BLE applications; added Zigbee_custom_ls_Client_Router, Zigbee_custom_ls_Server_Coord, Zigbee_OnOff_Client_Router_FreeRTOS, and Zigbee_OnOff_Server_Coord_FreeRTOS. |
| 07-Jul-2021 | 5 | Updated Table 1. STM32CubeWB firmware examples with the support for the Nucleo-WB15CC board<br>Updated the list of available projects:<br>• NUCLEO-WB15CC BLE_LLD Applications: BLE_LLD_Chat, BLE_LLD_Lowpower<br>• P-NUCLEO-WB55.Nucleo BLE_LLD Applications: BLE_LLD_Chat, BLE_LLD_Proximity<br>• P-NUCLEO-WB55.Nucleo Thread Applications: Thread_RCP<br>• STM32WB5MM-DK Demonstration Applications: Audio_BVLINKWB,BLE_HeartRate, BLE_Mesh_ThermometerSensor, BLE_Sensor, BLE_TransparentMode, BLE_p2pServer, BLE_LLD_Chat, Thread_Coap_Generic, Zigbee_TempMeas_Client_Router, Zigbee_TempMeas_Server_Coord, Zigbee_OnOff_Client_Router, Zigbee_OnOff_Server_Coord.<br>Removed BLE_Zigbee_Dyn_NVM application form the supported list. |
| 02-Dec-2021 | 6 | Updated Table 1. STM32CubeWB firmware examples:<br>• Added BLE_HeartRate_ota, BLE_MeshLightingLPN, BLE_MeshLightingPRFNode, BLE_Ota, BLE_p2pServer_ota, BLE_Mesh_Model_Sensor, BLE_AT_Server, BLE_LLD_Datarate, Thread_RCP_Cli_Cmd.<br>• Added Templates_LL for NUCLEO-WB15CC.<br>• Minor text edits across the whole table. |
| 30-Nov-2022 | 7 | Added support for the B-WB1M-WPAN1 board.<br>Updated Section 1 Reference documents.<br>Updated Table 1. STM32CubeWB firmware examples. |

# Contents