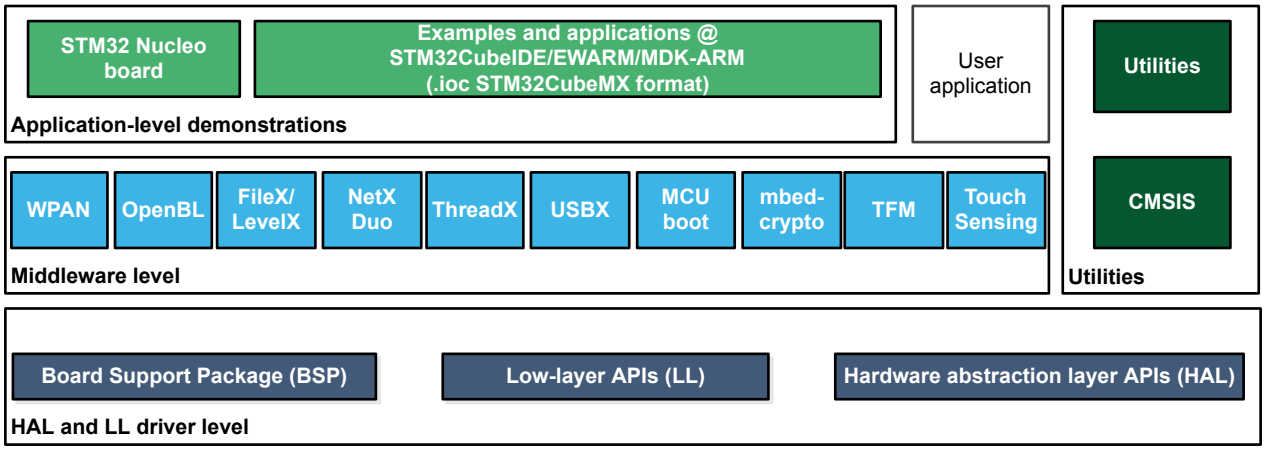
STM32Cube MCU Package examples for STM32WBA series

Introduction

The STM32CubeWBA 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. STM32CubeWBA firmware components



DT71823

1 STM32CubeWBA 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_WBA_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. STM32CubeWBA firmware examples contains the list of examples provided with STM32CubeWBA 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. STM32CubeWBA firmware examples

Level	Module name	Project name	Description	NUCLEO-WB52CG
Templates	-	TrustZoneDisabled	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application when security is not enabled (TZEN=0).	X
	-	TrustZoneEnabled	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application when TrustZone security is activated (option bit TZEN=1).	X TrustZone
	Total number of templates: 2			**2**
Templates_LL	-	TrustZoneDisabled	This project provides a reference template based on the STM32Cube LL API that can be used to build any firmware application	X
	Total number of templates_ll: 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
		ADC_MultiChannelSingleConversion	How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence.	MX
		ADC_Oversampling	How to use an ADC peripheral with oversampling.	MX
		ADC_SingleConversion_TriggerSW_IT	How to use an ADC peripheral to convert a single channel at each software start. The conversion is performed using the interrupt programming model.	MX
		ADC_SingleConversion_TriggerTimer_DMA	How to use an ADC peripheral to perform a single ADC conversion on a channel at each trigger event from a timer. Converted data are transferred by DMA into a table in RAM memory.	MX
	BSP	BSP_Example	This example describes how to use the BSP API.	MX
	CORTEX	CORTEXM_InterruptSwitch_TrustZone	How to first use an interrupt in secure application and later assign it to the nonsecure application when TrustZone security is activated (option bit TZEN=1).	MX TrustZone
		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
		CORTEXM_ProcessStack	How to modify the Thread mode stack. Thread mode is entered on reset, and can be entered as a result of an exception return.	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Examples	CORTEX	CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	MX
	CRC	CRC_Bytes_Stream_7bit_CRC	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes 7-bit CRC codes derived from buffers of 8-bit data (bytes). The user-defined generating polynomial is manually set to 0x65, that is, X^7 + X^6 + X^5 + X^2 + 1, as used in the Train Communication Network, IEC 60870-5[17].	MX
		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
	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 AES peripheral to encrypt and decrypt data using the AES 128 algorithm with ECB chaining mode in DMA mode.	MX
		CRYP_GCM_GMAC_CMAC_Suspension	How to use the CRYP AES peripheral to suspend then resume the AES GCM and GMAC CMAC processing of a message in order to carry out the encryption, decryption or authentication tag computation of a higher-priority message(CMAC).	MX
		CRYP_SAES_SharedKey	How to use the Secure AES coprocessor (SAES) peripheral to share application keys with the AES peripheral.	MX
		CRYP_SAES_WrapKey	How to use the Secure AES coprocessor (SAES) peripheral to wrap application keys using the hardware secret key DHUK then use it to encrypt in polling mode.	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
		DMA_MemToMem_TrustZone	How to use HAL DMA to perform memory to memory data transfers over secure and nonsecure DMA channels when TrustZone security is activated (option bit TZEN=1).	MX TrustZone
	FLASH	FLASH_BlockBased_TrustZone	How to configure and use the FLASH HAL API to managed block-based security of internal flash memory between secure and nonsecure applications when TrustZone security is activated (option bit TZEN=1).	MX TrustZone

Level	Module name	Project name	Description	NUCLEO-WB52CG
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
		FLASH_EraseProgram_TrustZone	How to configure and use the FLASH HAL API to erase and program the internal flash memory when TrustZone security is activated (option bit TZEN=1).	MX TrustZone
		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
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	MX
		GPIO_IOToggle_TrustZone	How to use HAL GPIO to toggle secure and unsecure IOs when TrustZone security is activated (option bit TZEN=1).	MX TrustZone
	GTZC	GTZC_TZSC_MPCBB_TrustZone	How to use HAL GTZC MPCBB to build any example with SecureFault detection when TrustZone security is activated (option bit TZEN=1).	MX TrustZone
	HAL	HAL_TimeBase	How to customize HAL using a general-purpose timer as main source of time base, instead of the Systick.	MX
		HAL_TimeBase_RTC_WKUP	How to customize the HAL using RTC wake-up as main source of time base, instead of Systick.	MX
		HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as main source of time base instead of Systick.	MX
	HASH	HASH_HMAC_SHA256MD5_IT_Suspension	How to suspend the HMAC digest computation when data are fed in interrupt mode.	MX
		HASH_SHA1MD5	This example provides a short description of how to use the HASH peripheral to hash data using SHA-1 and MD5 algorithms.	MX
		HASH_SHA1SHA224_IT_Suspension	How to suspend the HASH peripheral when data are fed in interrupt mode.	X
		HASH_SHA1_DMA_TrustZone	How to use a secure HASH SHA-1 computation service based on secure DMA channel when TrustZone security is activated (option bit TZEN=1).	MX TrustZone
		HASH_SHA224SHA256_DMA	How to use the HASH peripheral to hash data with SHA224 and SHA256 algorithms.	MX
	HSEM	HSEM_ProcessSync	How to use a hardware semaphore to synchronize two processes.	MX
		HSEM_ReadLock	How to enable, take, then release semaphore using two different processes.	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

Level	Module name	Project name	Description	NUCLEO-WB52CG
Examples	I2C	I2C_TwoBoards_ComDMA	How to handle I2C data buffer transmission/reception between two boards, via DMA.	MX
		I2C_TwoBoards_ComDMA_Autonomous_Master	How to handle I2C data buffer transmission/reception between two boards, via DMA.	MX
		I2C_TwoBoards_ComDMA_Autonomous_Slave	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
		I2C_TwoBoards_RestartAdvComIT	How to perform multiple I2C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	MX
		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
	IWDG	IWDG_WindowMode	How to update periodically the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset lap of time.	MX
	LPTIM	LPTIM_PulseCounter	How to configure and use, through the LPTIM HAL API, the LPTIM peripheral to count pulses.	MX
		LPTIM_Timeout	How to implement, through the HAL LPTIM API, a timeout with the LPTIMER peripheral, to wake up the system from a low-power mode.	MX
	PKA	PKA_ECCscalarMultiplication	How to use the PKA peripheral to execute ECC scalar multiplication. This allows generating 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
		PKA_ECDSA_Sign	How to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA).	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
		PKA_ModularExponentiation	How to use the PKA peripheral to execute modular exponentiation. This allows ciphering/deciphering a text.	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

Level	Module name	Project name	Description	NUCLEO-WB52CG
Examples	PKA	PKA_ModularExponentiation_IT	How to use the PKA peripheral to execute modular exponentiation. This allows ciphering/deciphering a text in interrupt mode.	MX
		PKA_PointCheck	How to use the PKA peripheral to determine if a point is on a curve. This allows validating 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_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	MX
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using an external reset or the WKUP pin.	MX
		PWR_STOP1	This example shows how to enter Stop 1 mode and wake up from this mode using an interrupt.	MX
	RAMCFG	RAMCFG_Parity_Error	How to configure and use the RAMCFG HAL API to enable parity error detection and generate parity error interruption.	MX
		RAMCFG_WriteProtection	How to configure and use the RAMCFG HAL API to configure RAMCFG SRAM write protection page.	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
		RCC_ClockConfig_TrustZone	How to configure the system clock (SYSCLK) in Run mode from the secure application upon request from the nonsecure application, using the RCC HAL API when TrustZone security is activated (option bit TZEN=1).	MX TrustZone
		RCC_LSEConfig	How to enable/disable the low-speed external(LSE) RC oscillator (about 32 kHz) in run time, using the RCC HAL API.	MX
		RCC_LSIConfig	How to enable/disable the low-speed internal (LSI) RC oscillator (about 32 kHz) in run time, using the RCC HAL API.	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
		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
		RTC_Calendar	How to configure the calendar using the RTC HAL API.	MX
		RTC_LSI	How to use of the LSI clock source autocalibration to get a precise RTC clock.	MX
		RTC_Tamper	How to configure the tamper detection with backup registers erase.	MX
		RTC_TimeStamp	How to configure the RTC HAL API to demonstrate the timestamp feature.	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Examples	RTC	RTC_TrustZone	How to configure the TrustZone-aware RTC peripheral when TrustZone security is activated (option bit TZEN=1): some features of the RTC can be secure while the others are non-secure.	MX TrustZone
	SPI	SPI_FullDuplex_ComDMA_Autonomous_Master	Data buffer transmission/reception between two boards via SPI using DMA.	MX
		SPI_FullDuplex_ComDMA_Autonomous_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	MX
		SPI_FullDuplex_ComDMA_Master	Data buffer transmission/reception between two boards via SPI using DMA.	MX
		SPI_FullDuplex_ComDMA_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	MX
		SPI_FullDuplex_ComIT_Master	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	MX
		SPI_FullDuplex_ComIT_Slave	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	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_ExtTriggerSynchro	This example shows how to synchronize TIM peripherals in cascade mode with an external trigger.	MX
		TIM_InputCapture	How to use the TIM peripheral to measure an external signal frequency.	MX
		TIM_OCActive	Configuration of the TIM 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
		TIM_OCInactive	Configuration of the TIM peripheral in Output Compare Inactive mode with the corresponding Interrupt requests for each channel.	MX
		TIM_OCToggle	Configuration of the TIM peripheral to generate four different signals at four different frequencies.	MX
		TIM_OnePulse	This example shows how to use the timer peripheral to generate a single pulse when a rising edge of an external signal is received on the timer input pin.	MX
		TIM_PWMInput	How to use the TIM peripheral to measure the frequency and duty cycle of an external signal.	MX
		TIM_PWMOutput	This example shows how to configure the TIM peripheral in PWM (Pulse Width Modulation) mode.	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
	UART	UART_HyperTerminal_DMA	UART transmission (transmit/receive) in DMA mode between a board and a HyperTerminal PC application.	MX
		UART_HyperTerminal_IT	UART transmission (transmit/receive) in Interrupt mode between a board and a HyperTerminal PC application.	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Examples	UART	UART_Printf	Rerouting of the C library printf function to the UART.	MX
	USART	USART_SlaveMode	This example describes a USART-SPI communication (transmit/receive) between two boards where the USART is configured as a slave.	MX
	WWDG	WWDG_Example	Configuration of the HAL API to update periodically the WWDG counter and simulate a software fault that generates an MCU WWDG reset when a predefined time period has elapsed.	MX
	Total number of examples: 103			**103**
Ewamples_LL	ADC	ADC_AnalogWatchdog_Init	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
		ADC_ContinuousConversion_TriggerSW_Init	How to use an ADC peripheral to convert a single channel continuously, from a software start.	MX
		ADC_Oversampling_Init	How to use an ADC peripheral with oversampling.	MX
		ADC_SingleConversion_TriggerSW_IT_Init	How to use ADC to convert a single channel at each software start, conversion performed using the interrupt programming model.	MX
		ADC_SingleConversion_TriggerSW_Init	How to use ADC to convert a single channel at each software start, conversion performed using the polling programming model.	MX
		ADC_TemperatureSensor_Init	How to use an ADC peripheral to perform a single ADC conversion on the internal temperature sensor and calculate the temperature in degrees Celsius.	MX
	CORTEX	CORTEX_MPU	The 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
	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).	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.	MX
	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 init usage.	MX
	EXTI	EXTI_ToggleLedOnIT_Init	This example describes 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 STM32WBAxx LL API. Peripheral initialization is done using the LL initialization function to demonstrate LL init usage.	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 STM32WBAxx LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Ewamples_LL	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
	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 for performance and size.	MX
		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 init 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 for 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 for performance and size.	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 for performance and size.	MX
		I2C_TwoBoards_MasterTx_SlaveRx_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 for performance and size.	MX
	LPTIM	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 STM32WBAxx LPTIM LL API. The peripheral is initialized with 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
	PWR	PWR_EnterStopMode	How to enter the Stop 0 mode.	MX
	RCC	RCC_OutputSystemClockOnMCO	How to configure the MCO pin (PA8) to output the system clock.	MX
		RCC_UseHSEasSystemClock	How to use of 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
	RNG	RNG_GenerateRandomNumbers	How to configure the RNG to generate 32-bit long random numbers.	MX
		RNG_GenerateRandomNumbers_IT	How to configure the RNG to generate 32-bit long random numbers using interrupts.	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Ewamples_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 periodically enter and wake up from Standby mode thanks to the RTC Wakeup Timer (WUT).	MX
		RTC_Tamper_Init	Configuration of 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	Configuration of 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_IT_Init	Configuration of 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 STM32WBAxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
	TIM	TIM_BreakAndDeadtime_Init	Configuration of the TIM peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined deadtime value, use the break feature, and lock the break and dead-time configuration.	MX
		TIM_InputCapture_Init	Use of the TIM 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 STM32WBAxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		TIM_OutputCompare_Init	How to configure the TIM peripheral to generate an output waveform in different output compare modes. This example is based on the STM32WBAxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		TIM_PWMOutput_Init	How to use a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32WBAxx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL Init usage.	MX
		TIM_TimeBase_Init	How to configure the TIM peripheral to generate a timebase. This example is based on the STM32WBAxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
	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 purpose (performance and size).	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Ewamples_LL	USART	USART_Communication_Rx_IT_Continuous_VCP_Init	This example shows how to configure the GPIO and USART peripherals for continuously receiving characters from HyperTerminal (PC) in Asynchronous mode using the Interrupt mode. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX
		USART_Communication_Rx_IT_Init	This example shows how to configure the GPIO and USART peripherals for receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL initialization function to demonstrate LL init usage.	MX
		USART_Communication_Rx_IT_VCP_Init	This example shows how to configure the GPIO and USART peripherals for receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL initialization function to demonstrate LL init usage.	MX
		USART_Communication_Tx_IT_Init	This example shows how to configure the GPIO and USART peripherals to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on STM32WBAxx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX
		USART_Communication_Tx_IT_VCP_Init	This example shows how to configure the GPIO and USART peripherals to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on the STM32WBAxx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX
		USART_Communication_Tx_Init	This example shows how to configure GPIO and USART peripherals to send characters asynchronously to a 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 STM32WBAxx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX
		USART_Communication_Tx_VCP_Init	This example shows how to configure GPIO and USART peripherals to send characters asynchronously to a 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 STM32WBAxx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX
		USART_HardwareFlowControl_Init	How to configure the GPIO and peripheral to receive characters asynchronously from a HyperTerminal (PC) in Interrupt mode with the hardware flow control feature enabled. This example is based on STM32WBAxx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Ewamples_LL	WWDG	WWDG_RefreshUntilUserEvent_Init	How to configure the WWDG to update periodically 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
	Total number of examples_ll: 49			**50**
Examples_MIX	ADC	ADC_SingleConversion_TriggerSW_IT	How to use ADC to convert a single channel at each SW start, conversion performed using the interrupt programming model.	MX
	CRC	CRC_PolynomialUpdate	How to use the CRC peripheral through the STM32WBAxx 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 STM32WBAxx DMA HAL and LL API. The LL API is used for performance improvement.	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 STM32WBAxx I2C HAL and LL API (LL API usage for performance improvement) and an interrupt.	MX
	PWR	PWR_STOP1	How to enter the Stop 1 mode and wake up from this mode by using the external reset or wake-up interrupt (all the RCC function calls use RCC LL API for minimizing footprint and maximizing performance).	MX
	TIM	TIM_PWMInput	How to use the TIM peripheral to measure an external signal frequency and duty cycle.	MX
	UART	UART_HyperTerminal_IT	How to use a UART to transmit data (transmit/receive) between a board and a HyperTerminal PC application in Interrupt mode. This example describes how to use the USART peripheral through the STM32WBAxx UART HAL and LL API, the LL API being used for performance improvement.	MX
		UART_HyperTerminal_TxPolling_RxIT	How to use a UART to transmit data (transmit/receive) between a board and a HyperTerminal PC application both in Polling and Interrupt modes. This example describes how to use the USART peripheral through the STM32WBAxx UART HAL and LL API, the LL API being used for performance improvement.	MX
	Total number of examples_mix: 8			**8**
Applications	-	OpenBootloader	This application exploits OpenBootloader middleware to demonstrate how to develop an IAP application and how to use it.	X
		SBSFU	The SBSFU provides a Root of Trust solution including Secure Boot and Secure Firmware Update functionalities that is used before executing the application and provides an example of secure service (GPIO toggle) that is isolated from the nonsecure application but can be used by the nonsecure application at run-time.	X
		TFM	The TFM provides a Root of Trust solution including Secure Boot and Secure Firmware Update functionalities that is used before executing the application and provides TFM secure services that are isolated from the non-secure application but can be used by the non-secure application at run-time.	X

Level	Module name	Project name	Description	NUCLEO-WB52CG
Applications	BLE	BLE_ApplicationInstallManager	The BLE_ApplicationInstallManager application, associated to a BLE application embedding OTA service, manages the firmware update over the air of a BLE application.	X
		BLE_Beacon	How to advertise 4 types of beacon (tlm, uuid, url, iBeacon).	MX
		BLE_DataThroughput_Client	How to demonstrate Point-to-Point communication using BLE component (as GATT server or GATT client).	MX
		BLE_DataThroughput_Server	How to demonstrate Point-to-Point communication using BLE component (as GATT server or GATT client).	MX
		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
		BLE_HeartRateThreadX	How to use the Heart Rate profile as specified by the BLE SIG with ThreadX OS.	MX
		BLE_HeartRate_ota	How to use the Heart Rate and OTA profile as specified by the BLE SIG.	MX
		BLE_SerialCom_Central	How to demonstrate Point-to-Point communication using BLE component.	MX
		BLE_SerialCom_Peripheral	How to demonstrate Point-to-Point communication using BLE component.	MX
		BLE_TransparentMode	How to communicate with the STM32CubeMonitor-RF tool using the transparent mode.	MX
		BLE_p2pClient	How to demonstrate STM32WBA acting as BLE central and GATT client.	MX
		BLE_p2pClient_Ext	How to demonstrate a BLE scanner with connections from an extended and a legacy advertising.	MX
		BLE_p2pRouter	How to demonstrate STM32WBA acting at the same time as both: BLE central and peripheral, GATT server and client.	MX
		BLE_p2pServer	How to demonstrate point-to-point communication using BLE (as GATT server).	MX
		BLE_p2pServerThreadX	How to demonstrate point-to-point communication using BLE (as GATT server).	MX
		BLE_p2pServer_Ext	How to demonstrate STM32WBA using BLE stack full version to use several extended advertising sets.	MX
		BLE_p2pServer_ota	How to demonstrate STM32WBA acting as BLE peripheral and GATT server.	MX
	FileX	Fx_File_Edit_Standalone	This application provides an example of FileX stack usage, running in standalone mode (without ThreadX). It demonstrates how to create a Fat File system on the internal SRAM memory using FileX.	MX
	LPM	Tiny_lpm_3modes	This example is based on the tiny lpm utility. It shows how the sequencer handles the task waiting for an event.	MX

Level	Module name	Project name	Description	NUCLEO-WB52CG
Applications	Sequencer	Sequencer_gpio_toggle	This example is based on the sequencer utilities. It shows how to use a sequencer task to toggle a GPIO.	MX
		Sequencer_gpio_toggle_lowpower	This example is based on the sequencer utilities.	MX
		Sequencer_task_pauseresume	This example is based on the sequencer utilities. It shows how the sequencer handles the task pause/ resume mechanism.	MX
		Sequencer_task_prio	This example is based on the sequencer utility. It shows how the sequencer manages task priority.	MX
		Sequencer_task_waitevent	This example is based on the sequencer utility. It shows how the sequencer handles the task waiting for an event.	MX
	ThreadX	Tx_CMSIS_Wrapper	This application provides an example of CMSIS RTOS adaptation layer for Azure RTOS ThreadX, it shows how to develop an application using the CMSIS RTOS 2 APIs.	X
		Tx_FreeRTOS_Wrapper	This application provides an example of Azure RTOS ThreadX stack usage, it shows how to develop an application using the FreeRTOS adaptation layer for ThreadX.	X
		Tx_LowPower	This application demonstrates how to configure ThreadX to operate in low-power mode. The system spends most of its time in Stop mode, and only wakes up when an external interrupt is triggered. User button B1 is the interrupt source.	MX
		Tx_MPU	This application provides an example of Azure RTOS ThreadX stack usage, it shows how to develop an application using the ThreadX module feature.	X
		Tx_SecureLEDToggle_TrustZone	This application provides an example of Azure RTOS ThreadX stack usage. It shows how to develop an application using the ThreadX when the TrustZone feature is enabled (TZEN=1).	MX TrustZone
		Tx_Thread_Creation	This application provides an example of Azure RTOS ThreadX stack usage, it shows how to develop an application using the ThreadX thread management APIs.	MX
		Tx_Thread_MsgQueue	This application provides an example of Azure RTOS ThreadX stack usage, it shows how to develop an application using the ThreadX message queue APIs. It demonstrates how to send and receive messages between threads using ThreadX message queue APIs.	MX
		Tx_Thread_Sync	This application provides an example of Azure RTOS ThreadX stack usage, it shows how to develop an application using the ThreadX synchronization APIs.	MX
	Total number of applications: 36			**36**
Total number of applications: 199				**199**

2 Reference documents

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

- Latest release of STM32CubeWBA firmware package
- *Getting started with STM32CubeWBA for STM32WBA series* (UM3131)

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

Revision history

Table 2. **Document revision history**

Date	Version	Changes
6-Mar-2023	1	Initial release.

Contents