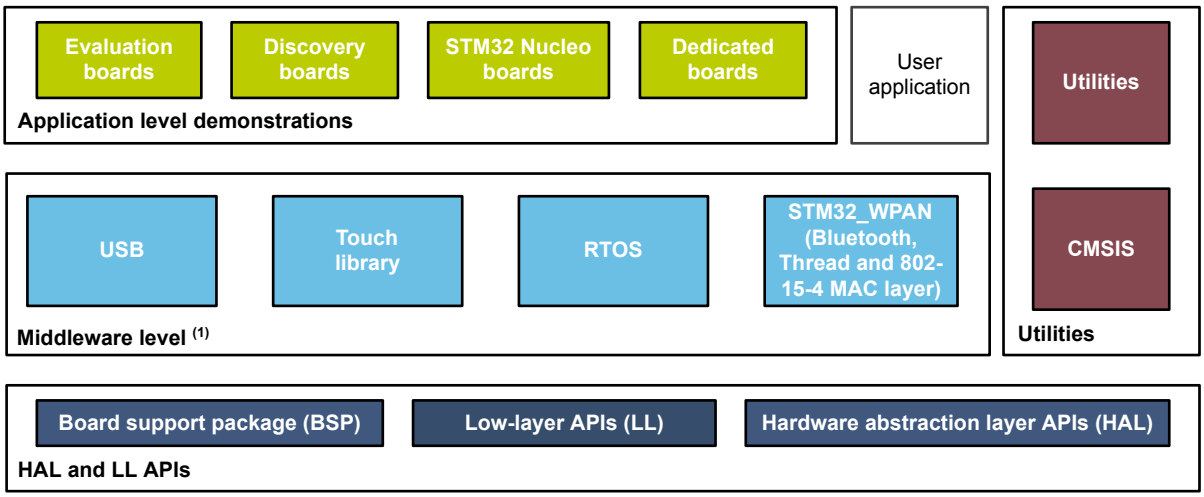
STM32Cube MCU Package examples for STM32WB Series

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.

MSv46868V1

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* (UM2250)
- *Description of STM32WB HAL and LL drivers* (UM2442)
- *Developing applications on STM32Cube with FatFS* (UM1721)
- *Developing applications on STM32Cube with RTOS* (UM1722)
- *Building a wireless application* (AN5289)

The microcontrollers of the STM32WB Series are based on Arm® Cortex® cores.

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, e.g. USB Host or FreeRTOS™) 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_STM32CubeWB_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 \SW4STM32* 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. STM32CubeWB firmware examples contains the list of examples provided with STM32CubeWB MCU Package.

The ***CubeMX*** label means that the projects have been created using STM32CubeMX, the STM32Cube™ initialization code generator. These projects can be open and modified with the STM32CubeMX tool. The others projects are manually created to demonstrate the product features.

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

Table 1. STM32CubeWB firmware examples

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Templates	-	Starter project	This projects provides a reference template that can be used to build any firmware application.	-	CubeMx
	Total number of templates: 1			0	1
Templates_LL	-	Starter project	This projects provides a reference template through the LL API that can be used to build any firmware application.	-	CubeMx
	Total number of templates_ll: 1			0	1
Examples	-	BSP	How to use the BSP API of the P-NUCLEO-WB55.USBDongle board.	X	-
	ADC	ADC_AnalogWatchdog	How to use the ADC to perform conversions with an analog watchdog and out-of-window interrupts enabled.	-	CubeMx
		ADC_MultiChannelSingleConversion	How to use the ADC to convert several channels using a sequencer in Discontinuous mode. Data conversions are indefinitely transferred by DMA into an array (circular mode).	-	CubeMx
		ADC_Oversampling	How to use the ADC to convert a single channel using the oversampling feature to increase resolution.	-	CubeMx
		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.	-	CubeMx
		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).	-	CubeMx
	BSP	BSP_Example	This example describes how to use the BSP API.	-	CubeMx
	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 (V_{REFINT}).	-	CubeMx
		COMP_CompareGpioVsVrefInt_Window_IT	How to make an analog watchdog using the COMP peripheral in Window mode.	-	CubeMx
	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).	-	CubeMx
		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.	-	CubeMx
	CRYP	CRYP_AESModes	How to use the CRYP peripheral to encrypt and decrypt data using AES in chaining modes (ECB, CBC, CTR).	-	CubeMx
		CRYP_DMA	How to use the AES1 peripheral to encrypt and decrypt data using AES 128 algorithm with ECB chaining mode in DMA mode.	-	CubeMx
	Cortex	CORTEXM_MPU	Presentation of the MPU feature. This example configures a memory area as privileged read-only and attempts to perform read and write operations in different modes.	-	CubeMx
		CORTEXM_ModePrivilege	How to modify the Thread mode privilege access and stack. The Thread mode is entered on reset or when returning from an exception.	-	CubeMx
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the HAL API.	-	CubeMx
		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.	-	CubeMx
		DMA_MUX_RequestGen	How to use the DMA with the DMAMUX request generator to generate DMA transfer requests upon an external line 4 rising edge signal.	-	CubeMx
	FLASH	FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal Flash memory.	-	CubeMx
		FLASH_WriteProtection	How to configure and use the FLASH HAL API to enable and disable the write protection of the internal Flash memory.	-	CubeMx
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	-	CubeMx
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	-	CubeMx
	HAL	HAL_TimeBase	How to customize HAL using a general-purpose timer as main timebase source, instead of Systick.	-	CubeMx
		HAL_TimeBase_RTC_ALARM	How to customize HAL using RTC alarm as main timebase source, instead of SysTick.	-	CubeMx
		HAL_TimeBase_RTC_WKUP	How to customize HAL using RTC wakeup as main timebase source, instead of SysTick.	-	CubeMx
		HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as main timebase source, instead of SysTick.	-	CubeMx
	HSEM	HSEM_ProcessSync	How to use a hardware semaphore to synchronize two processes.	-	CubeMx
		HSEM_ReadLock	How to enable, take, then release semaphore using two different processes.	-	CubeMx
	I2C	I2C_TwoBoards_AdvComIT	How to handle multiple I2C data buffer transmission/reception between two boards, using an interrupt.	-	CubeMx
		I2C_TwoBoards_ComDMA	How to handle I2C data buffer transmission/reception between two boards, via DMA.	-	CubeMx
		I2C_TwoBoards_ComIT	How to handle single I2C data buffer transmission/reception between two boards, using an interrupt.	-	CubeMx
		I2C_TwoBoards_ComPolling	How to handle I2C data buffer transmission/reception between two boards, in Polling mode.	-	CubeMx
		I2C_TwoBoards_RestartAdvComIT	How to perform multiple I2C data buffer transmission/reception between two boards, in Interrupt mode and using a restart condition.	-	CubeMx
		I2C_TwoBoards_RestartComIT	How to handle single I2C data buffer transmission/reception between two boards, in Interrupt mode and using a restart condition.	-	CubeMx
		I2C_WakeUpFromStop	How to handle I2C data buffer transmission/reception between two boards, using an interrupt when the device is in Stop mode.	-	CubeMx
		I2C_WakeUpFromStop2	How to handle I2C data buffer transmission/reception between two boards, using an interrupt when the device is in Stop 2 mode.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples	IWDG	IWDG_Reset	How to handle the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset when a programmed time period has elapsed.	-	CubeMx
		IWDG_WindowMode	How to periodically update the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset when a programmed time period has elapsed.	-	CubeMx
	LPTIM	LPTIM_PWMExternalClock	How to configure and use the LPTIM peripheral to generate a PWM signal at the lowest power consumption, using an external counter clock, through the HAL LPTIM API.	-	CubeMx
		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.	-	CubeMx
		LPTIM_PulseCounter	How to configure and use the LPTIM peripheral to count pulses, through the LPTIM HAL API.	-	CubeMx
		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.	-	CubeMx
	PKA	PKA_ECCscalarMultiplication	How to use the PKA peripheral to execute ECC scalar multiplications. This example allows the generation of a public key from a private key.	-	CubeMx
		PKA_ECCscalarMultiplication_IT	How to use the PKA peripheral to execute ECC scalar multiplications. This example allows the generation of a public key from a private key in Interrupt mode.	-	CubeMx
		PKA_ECDSA_Sign	How to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA).	-	CubeMx
		PKA_ECDSA_Sign_IT	How to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA) in Interrupt mode.	-	CubeMx
		PKA_ECDSA_Verify	How to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA).	-	CubeMx
		PKA_ECDSA_Verify_IT	How to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA) in Interrupt mode.	-	CubeMx
		PKA_ModularExponentiation	How to use the PKA peripheral to execute modular exponentiation. This example allows text ciphering/deciphering.	-	CubeMx
		PKA_ModularExponentiationCRT	How to compute the Chinese Remainder Theorem (CRT) optimization.	-	CubeMx
		PKA_ModularExponentiationCRT_IT	How to compute the Chinese Remainder Theorem (CRT) optimization in Interrupt mode.	-	CubeMx
		PKA_ModularExponentiation_IT	How to use the PKA peripheral to execute modular exponentiation. This allows text ciphering/deciphering in Interrupt mode.	-	CubeMx
		PKA_PointCheck	How to use the PKA peripheral to determine if a point is on a curve. This allows the validation an external public key.	-	CubeMx
		PKA_PointCheck_IT	How to use the PKA peripheral to determine if a point is on a curve. This example allows the validation of an external public key.	-	CubeMx
	PWR	PWR_LPRUN	How to enter and exit Low-power run mode.	-	CubeMx
		PWR_LPSLEEP	How to enter Low-power sleep mode and wake up from this mode using an interrupt.	-	CubeMx
		PWR_PVD	How to configure the programmable voltage detector using an external interrupt line. External DC supply must be used to supply V_{DD}.	-	CubeMx
		PWR_STANDBY_RTC	How to enter Standby mode and wake up from this mode using an external reset or the RTC Wakeup timer.	-	CubeMx
		PWR_STOP2_RTC	How to enter Stop 2 mode and wake up from this mode using an external reset or RTC Wakeup timer.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples	RCC	RCC_CRS_Synchronization_IT	How to configure the clock recovery service (CRS) in Interrupt mode, using the RCC HAL API.	-	CubeMx
		RCC_CRS_Synchronization_Polling	How to configure the clock recovery service (CRS) in Polling mode, using the RCC HAL API.	-	CubeMx
		RCC_ClockConfig	How to configure the system clock (SYSCLK) and modify the clock settings in Run mode, using the RCC HAL API.	-	CubeMx
	RNG	RNG_MultiRNG	How to configure the RNG using the HAL API. This example uses the RNG to generate 32-bit long random numbers.	-	CubeMx
		RNG_MultiRNG_IT	How to configure the RNG using the HAL API. This example uses RNG interrupts to generate 32-bit long random numbers.	-	CubeMx
	RTC	RTC_Alarm	How to configure and generate a RTC alarm using the RTC HAL API.	-	CubeMx
		RTC_Calendar	How to configure the calendar using the RTC HAL API.	-	CubeMx
		RTC_LSI	How to use the LSI clock source autocalibration to get a precise RTC clock.	-	CubeMx
		RTC_Tamper	How to configure the RTC HAL API to write/read data to/from RTC Backup registers.	-	CubeMx
		RTC_TimeStamp	How to configure the RTC HAL API to demonstrate the timestamp feature.	-	CubeMx
	SPI	SPI_FullDuplex_ComDMA_Master	Data buffer transmission/reception between two boards via SPI using DMA.	-	CubeMx
		SPI_FullDuplex_ComDMA_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	-	CubeMx
		SPI_FullDuplex_ComIT_Master	Data buffer transmission/reception between two boards via SPI in Interrupt mode.	-	CubeMx
		SPI_FullDuplex_ComIT_Slave	Data buffer transmission/reception between two boards via SPI in Interrupt mode.	-	CubeMx
		SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI in Polling mode.	-	CubeMx
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI in Polling mode.	-	CubeMx
	TIM	TIM_DMA	How to use the DMA with TIMER update request to transfer data from memory to TIMER Capture Compare Register 3 (TIMx_CCR3).	-	CubeMx
		TIM_DMABurst	How to update the TIMER channel 1 period and duty cycle using the TIMER DMA burst feature.	-	CubeMx
		TIM_InputCapture	How to use the TIM peripheral to measure an external signal frequency.	-	CubeMx
		TIM_OCActive	How to configure 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).	-	CubeMx
		TIM_OCInactive	How to configure the TIM peripheral in Output compare inactive mode with the corresponding interrupt requests for each channel.	-	CubeMx
		TIM_OCToggle	How to configure the TIM peripheral to generate four different signals at four different frequencies.	-	CubeMx
		TIM_OnePulse	How to use of the TIM 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 TIM peripheral to measure the frequency and duty cycle of an external signal.	-	CubeMx
		TIM_PWMOutput	How to configure the TIM peripheral in PWM (pulse width modulation) mode.	-	CubeMx
		TIM_TimeBase	How to configure the TIM peripheral to generate a timebase of 1 second with the corresponding interrupt request.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples	UART	UART_HyperTerminal_DMA	UART transmission (transmit/receive) in DMA mode between a board and an HyperTerminal PC application.	-	CubeMx
		UART_HyperTerminal_IT	UART transmission (transmit/receive) in Interrupt mode between a board and an HyperTerminal PC application.	-	CubeMx
		UART_Printf	Re-routing of the C library printf function to the UART.	-	CubeMx
		UART_TwoBoards_ComDMA	UART transmission (transmit/receive) in DMA mode between two boards.	-	CubeMx
		UART_TwoBoards_ComIT	UART transmission (transmit/receive) in Interrupt mode between two boards.	-	CubeMx
		UART_TwoBoards_ComPolling	UART transmission (transmit/receive) in Polling mode between two boards.	-	CubeMx
	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.	-	CubeMx
	Total number of examples: 93			1	92

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples_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.	-	CubeMx
		ADC_ContinuousConversion_TriggerSW	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	-	X
		ADC_ContinuousConversion_TriggerSW_Init	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	-	CubeMx
		ADC_ContinuousConversion_TriggerSW_LowPower_Init	How to use an ADC peripheral with ADC low-power features.	-	CubeMx
		ADC_GroupsRegularInjected_Init	How to use an ADC peripheral with both ADC groups (regular and injected) in their intended use cases.	-	CubeMx
		ADC_Oversampling_Init	How to use an ADC peripheral with ADC oversampling.	-	CubeMx
		ADC_SingleConversion_TriggerSW_DMA_Init	How to use an ADC peripheral to perform a single ADC 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).	-	CubeMx
		ADC_SingleConversion_TriggerSW_IT_Init	How to use an ADC peripheral to perform a single ADC 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).	-	CubeMx
		ADC_SingleConversion_TriggerSW_Init	How to use an ADC peripheral to perform a single ADC 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).	-	CubeMx
		ADC_SingleConversion_TriggerTimer_DMA_Init	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 indefinitely transferred by DMA into a table (circular mode).	-	CubeMx
		ADC_TemperatureSensor	How to use an ADC peripheral to perform a single ADC conversion on the internal temperature sensor and calculate the temperature in Celsius degrees.	-	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 init function to demonstrate LL initialization.	-	CubeMx
		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.	-	CubeMx
		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).	-	CubeMx
	CORTEX	CORTEX_MPU	Presentation of the MPU feature. This example configures a memory area as privileged read-only and attempts to perform read and write operations in different modes.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples_LL	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).	-	CubeMx
		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).	-	CubeMx
	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).	-	CubeMx
		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).	-	CubeMx
	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
		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 init functions to demonstrate LL initialization.	-	CubeMx
	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. This example 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	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 STM32WBxx LL API. The peripheral initialization is done using LL init function to demonstrate LL initialization.	-	CubeMx
	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 for 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 init function to demonstrate LL initialization.	-	CubeMx
	HSEM	HSEM_DualProcess	How to use the low-layer HSEM API to initialize, lock, and unlock hardware semaphore when two processes access the same resource.	-	CubeMx
		HSEM_DualProcess_IT	How to use the low-layer HSEM API to initialize, lock, and unlock hardware semaphore when two processes access the same resource.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples_LL	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 for performance and size.	-	CubeMx
		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.	-	CubeMx
		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 for 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 init function to demonstrate LL initialization.	-	CubeMx
		I2C_OneBoard_Communication_PollingAndIT_Init	How to transmit data bytes from an I2C master device in Polling mode to an I2C slave device in Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	CubeMx
		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.	-	CubeMx
		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.	-	CubeMx
		I2C_TwoBoards_MasterTx_SlaveRx_Init	How to transmit data bytes from an I2C master device in 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.	-	CubeMx
		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 for performance and size.	-	CubeMx
		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 for performance and size.	-	CubeMx
	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 for performance and size.	-	CubeMx
	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 for 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 init function to demonstrate LL initialization.	-	CubeMx
	LPUART	LPUART_WakeUpFromStop2_Init	How to configure the 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 init function to demonstrate LL initialization.	-	CubeMx
		LPUART_WakeUpFromStop_Init	How to configure the 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 init function to demonstrate LL initialization.	-	CubeMx
	PKA	PKA_ECDSA_Sign	How to use the low-layer PKA API to generate an ECDSA signature.	-	CubeMx
		PKA_ModularExponentiation	How to use the low-layer PKA API to execute RSA modular exponentiation.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
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.	-	CubeMx
		PWR_EnterStopMode	How to enter Stop 2 mode.	-	CubeMx
		PWR_OptimizedRunMode	How to increase/decrease frequency and V_{CORE} and enter/exit Low-power run mode.	-	CubeMx
		PWR_SMPS_16MHZ_HSI	This example shows how to use STM32WBxx power converters (SMPS, LDO and LP-LDO) depending on V_{DD} voltage and low-power mode.	-	CubeMx
		PWR_SMPS_64MHZ_MSI_PLL	This example shows how to use STM32WBxx power converters (SMPS, LDO and LP-LDO) depending on V_{DD} voltage and low-power mode.	-	CubeMx
	RCC	RCC_HWAutoMSICalibration	How to use the MSI clock source hardware auto-calibration and the LSE clock (PLL mode) to obtain an accurate MSI clock.	-	CubeMx
		RCC_OutputSystemClockOnMCO	How to configure MCO pin (PA8) to output the system clock.	-	CubeMx
		RCC_UseHSEasSystemClock	How to use the RCC LL API to start the HSE and use it as system clock.	-	CubeMx
		RCC_UseHSI_PLLasSystemClock	How to modify the PLL parameters in runtime.	-	CubeMx
	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).	-	CubeMx
		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).	-	CubeMx
	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
		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 init function.	-	CubeMx
		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).	-	CubeMx
		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).	-	CubeMx
		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).	-	CubeMx
		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).	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples_LL	SPI	SPI_OneBoard_HalfDuplex_DMA	How to configure the 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 the 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 init function to demonstrate LL initialization.	-	CubeMx
		SPI_OneBoard_HalfDuplex_IT_Init	How to configure the 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).	-	CubeMx
		SPI_TwoBoards_FullDuplex_DMA_Master_Init	How to transfer and receive a data buffer via SPI 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).	-	CubeMx
		SPI_TwoBoards_FullDuplex_DMA_Slave_Init	How to transfer and receive a data buffer via SPI 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).	-	CubeMx
		SPI_TwoBoards_FullDuplex_IT_Master_Init	How to transfer and receive a data buffer via SPI 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).	-	CubeMx
		SPI_TwoBoards_FullDuplex_IT_Slave_Init	How to transfer and receive a data buffer via SPI 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).	-	CubeMx
	TIM	TIM_BreakAndDeadtime	How to configure the TIM 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).	-	CubeMx
		TIM_InputCapture_Init	How to use 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 STM32WBxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	CubeMx
		TIM_OnePulse	How to configure a 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 TIM 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).	-	CubeMx
		TIM_PWMOutput	How to use a 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 a 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 init function to demonstrate LL initialization.	-	CubeMx
		TIM_TimeBase_Init	How to configure the TIM 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).	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Examples_LL	USART	USART_Communication_Rx_IT	How to configure the 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 the GPIO and USART peripherals to continuously receive characters from an HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization uses LL unitary services functions for optimization purpose (performance and size).	-	CubeMx
		USART_Communication_Rx_IT_Continuous_VCP_Init	How to configure the GPIO and USART peripherals to continuously receive characters from an HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	CubeMx
		USART_Communication_Rx_IT_Init	How to configure the GPIO and USART peripherals to receive characters from an HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization is done using LL init function to demonstrate LL initialization.	-	CubeMx
		USART_Communication_Rx_IT_VCP_Init	How to configure the GPIO and USART peripherals to receive characters from an HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization is done using LL init function to demonstrate LL initialization.	-	CubeMx
		USART_Communication_TxRx_DMA_Init	How to configure the GPIO and USART peripherals 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).	-	CubeMx
		USART_Communication_Tx_IT_Init	How to configure the GPIO and USART peripherals to send characters asynchronously to an 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).	-	CubeMx
		USART_Communication_Tx_IT_VCP_Init	How to configure the GPIO and USART peripherals to send characters asynchronously to an 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).	-	CubeMx
		USART_Communication_Tx_Init	How to configure the 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 allows to exit 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).	-	CubeMx
		USART_Communication_Tx_VCP_Init	How to configure the 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 allows to exit 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).	-	CubeMx
		USART_WakeUpFromStop1_Init	How to configure the GPIO and USART1 peripherals to allow the characters received on USART_RX pin to wake up the MCU from low-power mode.	-	CubeMx
		USART_WakeUpFromStop_Init	How to configure the GPIO and USART1 peripherals to allow the characters received on USART_RX pin to wake up the MCU from low-power mode.	-	CubeMx
	UTILS	UTILS_ConfigureSystemClock	How to use UTILS LL API to configure the system clock using the PLL with HSI as source clock.	-	CubeMx
		UTILS_ReadDeviceInfo	This example reads the UID, Device ID and Revision ID and saves them into a global information buffer.	-	CubeMx
	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).	-	CubeMx
	Total number of examples_ll: 92			0	92

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
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.	-	CubeMx
	CRC	CRC_PolynomialUpdate	How to use the CRC peripheral through the STM32WBxx CRC HAL and LL API.	-	CubeMx
	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.	-	CubeMx
	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 bits or 10 bits). This example uses the STM32WBxx I2C HAL and LL API (LL API usage for performance improvement) and an interrupt.	-	CubeMx
	PWR	PWR_STOP1	How to enter Stop 1 mode and wake up from this mode using an external reset or a wakeup interrupt. All the RCC function calls use RCC LL API for minimizing footprint and maximizing performance.	-	CubeMx
	SPI	SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI in Polling mode.	-	CubeMx
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI in Polling mode.	-	CubeMx
		SPI_HalfDuplex_ComPollingIT_Master	Data buffer transmission/reception between two boards via SPI in Polling (LL driver) and Interrupt modes (HAL driver).	-	CubeMx
		SPI_HalfDuplex_ComPollingIT_Slave	Data buffer transmission/reception between two boards via SPI in Polling (LL driver) and Interrupt modes (HAL driver).	-	CubeMx
	TIM	TIM_PWMInput	How to use the TIM peripheral to measure an external signal frequency and duty cycle.	-	CubeMx
	UART	UART_HyperTerminal_IT	How to use of 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.	-	CubeMx
		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.	-	CubeMx
	Total number of examples_mix: 12			0	12

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Applications	BLE	BLE_Beacon	How to advertise three types of beacons (tlm, uuid, url).	-	CubeMx
		BLE_BloodPressure	How to use the Blood Pressure profile as specified by the BLE SIG.	-	CubeMx
		BLE_CableReplacement	How to use the point-to-point communication using the BLE component.	-	X
		BLE_DataThroughput	How to use data throughput via notification from server to client using the BLE component.	-	X
		BLE_HealthThermometer	How to use the Health Thermometer profile as specified by the BLE SIG.	-	CubeMx
		BLE_HeartRate	How to use the Heart Rate profile as specified by the BLE SIG.	X	CubeMx
		BLE_HeartRateFreeRTOS	How to use the 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.	-	X
		BLE_Hid	How to use the Human Interface Device profile as specified by the BLE SIG.	-	X
		BLE_MeshLightingDemo	This application implements the BLE Mesh Lighting profile as specified by the BLE SIG.	-	X
		BLE_Ota	This application implements OTA to download a new image into the user Flash memory.	-	X
		BLE_Proximity	How to use the Proximity profile as specified by the BLE SIG.	-	X
		BLE_TransparentMode	How to communicate with the STM32CubeMonitor-RF (STM32CubeMonRF) tool in Transparent mode.	-	CubeMx
		BLE_TransparentModeVCP	How to communicate with the STM32CubeMonitor-RF (STM32CubeMonRF) tool in Transparent mode.	X	-
		BLE_p2pClient	This example demonstrates Point-to-Point communication using the BLE component.	X	CubeMx
		BLE_p2pRouteur	This example demonstrates Multipoint communication using the BLE component.	X	CubeMx
		BLE_p2pServer	This example demonstrates Point-to-Point communication using the BLE component.	X	CubeMx
		BLE_p2pServer_ota	This example demonstrates Point-to-Point communication using the BLE component.	-	X
	BLE_Thread	Ble_Thread_Static	How to use BLE and Thread® applications in Static concurrent mode.	-	X
	FatFs	FatFs_uSD_Standalone	How to use STM32CubeWB firmware with FatFS middleware component as a generic FAT file system module. This example develops an application that exploits FatFS features to configure a microSD drive.	-	CubeMx
	FreeRTOS	FreeRTOS_Mail	How to use mail queues with CMSIS RTOS API.	-	CubeMx
		FreeRTOS_Mutexes	How to use mutexes with CMSIS RTOS API.	-	CubeMx
		FreeRTOS_Queues	How to use message queues with CMSIS RTOS API.	-	CubeMx
		FreeRTOS_Semaphore	How to use semaphores with CMSIS RTOS API.	-	CubeMx
		FreeRTOS_SemaphoreFromISR	How to use semaphore from ISR with CMSIS RTOS API.	-	CubeMx
		FreeRTOS_Signal	How to perform Thread signaling using CMSIS RTOS API.	-	CubeMx
		FreeRTOS_SignalFromISR	This application shows the usage of CMSIS-OS Signal API from ISR context.	-	CubeMx
		FreeRTOS_ThreadCreation	How to implement thread creation using CMSIS RTOS API.	-	CubeMx
		FreeRTOS_Timers	How to use timers of CMSIS RTOS API.	-	CubeMx

Level	Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
Applications	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_RFD	How to use MAC 802.15.4 Association and Data exchange.	-	X
	Thread®	Thread_Cli_Cmd	How to control the Thread® stack via Cli commands.	X	CubeMx
		Thread_Coap_DataTransfer	How to transfer large blocks of data through the CoAP messaging protocol.	X	X
		Thread_Coap_Generic	How to build Thread® application based on Coap messages.	X	CubeMx
		Thread_Coap_MultiBoard	How to use Coap for sending message to multiple boards.	-	X
		Thread_Commissioning	How to use Thread® commissioning process.	-	X
		Thread_FTD_Coap_Multicast	How to exchange multicast Coap messages.	X	X
		Thread_SED_Coap_Multicast	How to exchange a Coap message using the Thread® protocol.	X	X
	USB_Device	CDC_Standalone	This application describes how to use USB Device application based on the Device Communication Class (CDC) following the PSTN subprotocol on the STM32WBxx devices.	-	CubeMx
		DFU_Standalone	Compliant implementation of the Device Firmware Upgrade (DFU) capability to program the embedded Flash memory through the USB peripheral.	-	CubeMx
		HID_Standalone	Use of the USB Device application based on the Human Interface (HID).	-	CubeMx
		MSC_Standalone	This application shows how to use the USB Device application based on the Mass Storage Class (MSC) on the STM32WB55xx devices.	-	CubeMx
	Total number of applications: 51			10	41
Demonstrations	-	Adafruit_LCD_1_8_SD_Joystick	This demonstration firmware is based on STM32Cube. It helps you to discover STM32 Cortex-M devices that can be plugged on a STM32 Nucleo board.	-	CubeMx
	Total number of demonstrations: 1			0	1
Total number of projects: 251				11	240

Revision history

Table 2. **Document revision history**

Date	Version	Changes
19-Feb-2019	1	Initial release.

Contents