AN5409

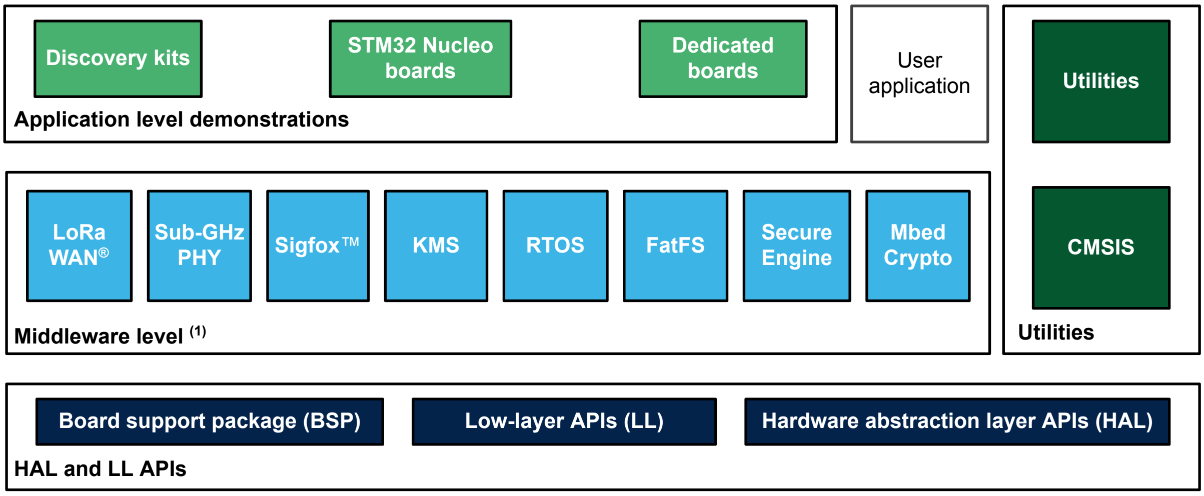
Application note

STM32Cube MCU Package examples for STM32WL Series

Introduction

The STM32CubeWL MCU Package is delivered 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.

Figure 1. STM32CubeWL firmware components



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

1 Reference documents

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

- Latest release of STM32CubeWL firmware package
- *Getting started with STM32CubeWL for STM32WL Series* (UM2643)
- *Description of STM32WL HAL drivers* (UM2642)
- *Developing applications on STM32Cube with FatFs* (UM1721)
- *Developing applications on STM32Cube with RTOS* (UM1722)
- *How to build a Sigfox™ application with STM32CubeWL* (AN5480)
- *How to build a LoRa® application with STM32CubeWL* (AN5406)
- *Getting started with the SBSFU of STM32CubeWL* (UM2767)
- *Getting started with STM32WL dual core using IAR™ and Keil®* (AN5556)

The microcontrollers of the STM32WL 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 STM32CubeWL examples

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

- **Examples**: these examples use only the HAL and BSP drivers (middleware components not used). Their objective is to demonstrate the product/peripherals features and usage. They are organized per peripheral (one folder per peripheral, for example TIM). Their complexity level ranges from the basic usage of a given peripheral (such as PWM generation using timer) to the integration of several peripherals (such as 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 use only the LL drivers (HAL and middleware components not used). They offer an optimum implementation of typical use cases of the peripheral features and configuration procedures. The examples are organized per peripheral (a folder for each peripheral, such as TIM).
- **Examples_MIX**: these examples use both HAL and LL drivers. They offer an optimum implementation of typical use cases of the peripheral features and configuration procedures. The examples are organized per peripheral (a folder for each peripheral, such as DMA).
- **Applications**: the applications demonstrate the product performance and how to use the available middleware stacks. They are organized by middleware (a folder per middleware, for example LoRaWAN®). 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.
- **HAL template projects**: the HAL template projects are provided to allow the user to quickly build a firmware application on a given board. They are provided both for single- and dual-core STM32WL microcontrollers.
- **LL template projects**: the LL template projects are provided to allow the user to quickly build a firmware application on a given board. They are provided both for single- and dual-core STM32WL microcontrollers.

The examples are located under *STM32Cube_FW_WL_VX.Y.Z\Projects*.

All the examples provided for single-core STM32WLEx microcontrollers have the same structure:

- *\Inc* folder, which contains all header files.
- \Src folder, which contains the sources code.
- *\EWARM*, *\MDK-ARM* and */STM32CubeIDE* folders, which contain the preconfigured project for each toolchain.
- readme.txt file, which describes the example behavior and the environment required to run the example.

All the examples provided for dual-core STM32WL5x microcontrollers have the same structure:

- *CM0PLUS\Inc* and *CM4\Inc* folders, which contain all header files for Arm® Cortex®-M0+ and Arm® Cortex®-M4, respectively.
- *CM0PLUS\Src* and *CM4\Src* folders, which contain the sources code for Arm® Cortex®-M0+ and Arm® Cortex®-M4, respectively.
- A *Common\ folder*, which contains the common files for Arm® Cortex®-M0+ and Arm® Cortex®-M4.
- *\EWARM*, *\MDK-ARM*, */STM32CubeIDE* and *\STM32CubeIDE* folders, which contain the preconfigured project for each toolchain.
- readme.txt file, which describes the example behavior and the environment required to run the example.

To run the example, proceed as follows:

1. Open the example using the 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 (such as LED, LCD display and push-buttons). The BSP is based on a modular architecture that can be easily ported to any hardware by implementing the low-level routines.

Table 1. STM32CubeWL firmware examples contains the list of examples provided with STM32CubeWL MCU Package.

Note: *STM32CubeMX-generated examples are highlighted with the* MX *STM32CubeMX icon.*

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

Table 1. STM32CubeWL firmware examples

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Templates	-	DualCore	This projects provides a reference template that can be used to build any dual-core (Arm® Cortex®-M4/-M0) firmware application.	MX
		SingleCore	This projects provides a reference template that can be used to build any single-core (Arm® Cortex®-M4) firmware application.	MX
	Total number of templates: 2			**2**
Templates_LL	-	DualCore	This projects provides a reference template through the LL API that can be used to build any dual-core (Arm® Cortex®-M4/-M0) firmware application.	MX
		SingleCore	This projects provides a reference template through the LL API that can be used to build any single-core (Arm® Cortex®-M4) firmware application.	MX
	Total number of templates_ll: 2			**2**
Examples	ADC	ADC_AnalogWatchdog	This example describes how to use the ADC peripheral to perform conversions with an analog watchdog and out-of-window interrupts enabled.	MX
		ADC_MultiChannelSingleConversion	This example describes 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
		ADC_Oversampling	This example describes how to use the ADC to convert a single channel using the oversampling feature to increase resolution.	MX
		ADC_SingleConversion_TriggerSW_IT	This example describes how to use the ADC to convert a single channel at each software start. The conversion is performed using the interrupt programming model. The ADC is configured to convert a single channel in Single conversion mode, starting from a software trigger.	MX
		ADC_SingleConversion_TriggerTimer_DMA	This example describes how to use the ADC to convert a single channel at each trigger from a timer. Converted data are indefinitely transferred by DMA into an array (Circular mode).	MX
	COMP	COMP_CompareGpioVsVrefInt_IT	This example describes how to configure the COMP peripheral to compare the external voltage applied on a specific pin with the internal voltage reference.	MX
		COMP_CompareGpioVsVrefInt_Window_IT	This example describes 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.	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	This example describes 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	This example describes how to modify the Thread mode stack. Thread mode is entered on reset and can be entered when returning from an exception.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples	CORTEX	CORTEXM_SysTick	This example describes how to use the default SysTick configuration with a 1 ms timebase, to toggle LEDs.	MX
	CRC	CRC_Bytes_Stream_7bit_CRC	This example describes 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	This example describes 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	This example describes 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	This example describes 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	This example describes how to use the CRYP peripheral to encrypt and decrypt data using AES in chaining modes (ECB, CBC, CTR).	MX
		CRYP_DMA	This example describes how to use the AES peripheral to encrypt and decrypt data using AES-128 algorithm with ECB chaining mode in DMA mode.	MX
	DAC	DAC_SignalsGeneration	This example describes how to use the DAC peripheral to generate several signals using the DMA controller and the DAC internal wave generator.	MX
		DAC_SimpleConversion	This example describes how to use the DAC peripheral to perform a simple conversion.	MX
	DMA	DMA_FLASHToRAM	This example describes how to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the HAL API.	MX
		DMA_MUXSYNC	This example describes 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
		DMA_MUX_RequestGen	This example describes how to use the DMA with the DMAMUX request generator to generate DMA transfer requests upon an external line 0 rising edge signal.	MX
		DMA_SecureTransfer_DualCore	This example describes how to configure the DMA to perform a data transfer from secured area to unsecured area.	MX
	FLASH	FLASH_EraseProgram	This example describes how to configure and use the FLASH HAL API to erase and program the internal Flash memory.	MX
		FLASH_FastProgram	This example describes how to configure and use the FLASH HAL API to erase and fast-program the internal Flash memory.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples	FLASH	FLASH_WriteProtection	This example describes 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	This example describes how to configure external interrupt lines.	MX
		GPIO_IOToggle	This example describes how to configure and use GPIOs through the HAL API.	MX
	GTZC	GTZC_GlobalSecurityConfiguration_DualCore	This example describes how to configure Flash option bytes and GTZC to fully secure all resources from unauthorized accesses.	X
		GTZC_MemoryWatermarkProtection_DualCore	This example shows how to protect a secure memory area from unprivileged accesses and set up illegal access notification.	MX
		GTZC_PeripheralProtection_DualCore	This example shows how to secure peripheral IP access and set up illegal access notification.	MX
	HAL	HAL_TimeBase	This example shows how to customize the HAL using a general-purpose timer as main timebase source, instead of the SysTick.	MX
		HAL_TimeBase_RTC_ALARM	This example shows how to customize the HAL using RTC alarm as main timebase source, instead of the SysTick.	MX
		HAL_TimeBase_RTC_WKUP	This example shows how to customize the HAL using RTC wakeup as main timebase source, instead of the SysTick.	MX
		HAL_TimeBase_TIM	This example shows how to customize the HAL using a general-purpose timer as main timebase source instead of the SysTick.	MX
	HSEM	HSEM_ProcessSync	This example describes how to use a hardware semaphore to synchronize two processes.	MX
		HSEM_ReadLock	This example shows how to enable, take, and then release a semaphore using two different processes.	MX
		HSEM_Sync_DualCore	This example shows how to synchronize two CPUs using the HSEM peripheral to safely access a shared resource.	MX
	I2C	I2C_TwoBoards_ComDMA	This example shows how to handle I2C data buffer transmission/reception between two boards, via DMA.	MX
		I2C_TwoBoards_ComIT	This example shows how to handle I2C data buffer transmission/reception between two boards, using an interrupt.	MX
		I2C_TwoBoards_ComPolling	This example shows how to handle I2C data buffer transmission/reception between two boards, in Polling mode.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples	I2C	I2C_TwoBoards_RestartAdvComIT	This example shows how to perform multiple I2C data buffer transmission/reception between two boards, in Interrupt mode and with a restart condition.	MX
		I2C_TwoBoards_RestartComIT	This example shows how to handle single I2C data buffer transmission/reception between two boards, in Interrupt mode and with a restart condition.	MX
		I2C_WakeUpFromStop	This example shows how to handle I2C data buffer transmission/reception between two boards, using an interrupt when the device is in Stop mode.	MX
		I2C_WakeUpFromStop2	This example shows how to handle I2C data buffer transmission/reception between two boards, using an interrupt when the device is in Stop2 mode.	MX
	IPCC	IPCC_HalfDuplexMode_DualCore	This example shows how to transfer data between two processors using IPCC Half-duplex channel mode.	MX
		IPCC_SimplexMode_DualCore	This example shows how to transfer data between two processors using IPCC Simplex channel mode.	MX
	IWDG	IWDG_Reset	This example shows how to handle the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset laps of time.	MX
		IWDG_WindowMode	This example shows how to periodically update the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset laps of time.	MX
	LPTIM	LPTIM_PWMExternalClock	This example shows how to configure and use the LPTIM peripheral to generate a PWM signal at the lowest power consumption, using an external counter clock and through the HAL LPTIM API.	MX
		LPTIM_PWM_LSE	This example shows how to configure and use the LPTIM peripheral to generate a PWM signal in low-power mode, using the LSE as counter clock and through the HAL LPTIM API.	MX
		LPTIM_PulseCounter	This example shows how to configure and use, through the LPTIM HAL API, the LPTIM peripheral to count pulses.	MX
		LPTIM_Timeout	This example shows how to implement a timeout with the LPTIM peripheral to wake up the system from low-power mode, through the HAL LPTIM API.	MX
	PKA	PKA_ECCscalarMultiplication	This example describes how to use the PKA peripheral to execute ECC scalar multiplication. This enables generating a public key from a private key.	MX
		PKA_ECCscalarMultiplication_IT	This example describes how to use the PKA peripheral to execute ECC scalar multiplication. This enables generating a public key from a private key in Interrupt mode.	MX
		PKA_ECDSA_Sign	This example shows how to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA).	MX
		PKA_ECDSA_Sign_IT	This example shows how to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA) in Interrupt mode.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples	PKA	PKA_ECDSA_Verify	This example shows how to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA).	MX
		PKA_ECDSA_Verify_IT	This example shows how to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA) in Interrupt mode.	MX
		PKA_ModularExponentiation	This example describes how to use the PKA peripheral to execute modular exponentiation. This enables ciphering/deciphering a text.	MX
		PKA_ModularExponentiationCRT	This example shows how to compute the Chinese Remainder Theorem (CRT) optimization.	MX
		PKA_ModularExponentiationCRT_IT	This example shows how to compute the Chinese Remainder Theorem (CRT) optimization in Interrupt mode.	MX
		PKA_ModularExponentiation_IT	This example describes how to use the PKA peripheral to execute modular exponentiation. This enables ciphering/deciphering a text in Interrupt mode.	MX
		PKA_PointCheck	This example describes how to use the PKA peripheral to determine if a point is on a curve. This enables validating an external public key.	MX
		PKA_PointCheck_IT	This example describes how to use the PKA peripheral to determine if a point is on a curve. This enables validating an external public key.	MX
	PWR	PWR_LPRUN	This example shows how to enter and exit Low-power run mode.	MX
		PWR_LPSLEEP	This example shows how to enter Low-power sleep mode and wake up from this mode by using an interrupt.	MX
		PWR_PVD	This example shows how to configure the programmable voltage detector by using an external interrupt line. External DC supply must be used to supply V_{DD}.	MX
		PWR_SMPS	This example shows how to configure and use the SMPS through the HAL API.	MX
		PWR_STANDBY	This example shows how to enter Standby mode and wake up from this mode by using an external reset or the WKUP pin.	MX
		PWR_STANDBY_RTC	This example shows how to enter Standby mode and wake up from this mode by using an external reset or the RTC wakeup timer.	MX
		PWR_STOP2_RTC	This example shows how to enter Stop2 mode and wake up from this mode using an external reset or RTC wakeup timer.	MX
		PWR_SecurityIllegalAccess_DualCore	This example shows how to manage illegal access in multicore program and low-power modes.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples	PWR	PWR_Standby_DualCore	This example shows how to manage system low-power Standby mode and each CPU low-power modes in multicore program mode.	MX
		PWR_Stop2_DualCore	This example shows how to manage system low-power Stop mode and each CPU low-power modes in multicore program mode.	MX
		PWR_Stop2_PeripheralNoRetention	This example describes how to use low-power Stop2 mode and peripherals that do not feature configuration retention.	MX
	RCC	RCC_ClockConfig	This example describes how to configure the system clock (SYSCLK) and modify the clock settings in Run mode, using the RCC HAL API.	MX
		RCC_LSEConfig	This example describes how to enable/disable the low-speed external (LSE) RC oscillator (about 32 KHz) at runtime, using the RCC HAL API.	MX
		RCC_LSIConfig	This example shows how to enable/disable the low-speed internal (LSI) RC oscillator (about 32 KHz) at runtime, using the RCC HAL API.	MX
	RNG	RNG_MultiRNG	This example shows 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	This example shows 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	This example shows how to configure and generate an RTC alarm using the RTC HAL API.	MX
		RTC_Alarm_DualCore	This multicore project shows how Arm® Cortex®-M4 core uses Alarm A, and Arm® Cortex®-M0 Alarm B.	MX
		RTC_Binary	This example shows how to configure the binary mode with binary Alarm A and read the time.	MX
		RTC_Calendar	This example shows how to configure the calendar using the RTC HAL API.	MX
		RTC_LSI	This example shows how to use the LSI clock source autocalibration to obtain an accurate RTC clock.	MX
		RTC_Tamper	This example shows how to configure the tamper detection with backup registers erase.	MX
		RTC_TimeStamp	This example shows how to configure the RTC HAL API to demonstrate the timestamp feature.	MX
	SPI	SPI_FullDuplex_ComDMA_Master	This example shows how to perform data buffer transmission/reception between two boards via SPI using DMA.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples	SPI	SPI_FullDuplex_ComDMA_Slave	This example shows how to perform data buffer transmission/reception between two boards via SPI using DMA.	MX
		SPI_FullDuplex_ComIT_Master	This example shows how to perform data buffer transmission/reception between two boards via SPI in Interrupt mode.	MX
		SPI_FullDuplex_ComIT_Slave	This example shows how to perform data buffer transmission/reception between two boards via SPI in Interrupt mode.	MX
		SPI_FullDuplex_ComPolling_Master	This example shows how to perform data buffer transmission/reception between two boards via SPI in Polling mode.	MX
		SPI_FullDuplex_ComPolling_Slave	This example shows how to perform data buffer transmission/reception between two boards via SPI in Polling mode.	MX
	SUBGHZ	SUBGHZ_Tx_Mode	This example shows how to configure the SUBGHZ peripheral to set the Sub-GHz radio in Tx mode.	MX
	TIM	TIM_DMA	This example shows how to use the DMA with timer update request to transfer data from memory to timer capture compare register 3 (TIMx_CCR3).	MX
		TIM_DMABurst	This example shows how to update the timer channel 1 period and duty cycle using the timer DMA burst feature.	MX
		TIM_InputCapture	This example describes how to use the TIM peripheral to measure an external signal frequency.	MX
		TIM_OCActive	This example describes 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).	MX
		TIM_OCInactive	This example describes how to configure the TIM peripheral in Output compare inactive mode with the corresponding interrupt requests for each channel.	MX
		TIM_OCToggle	This example describes how to configure the TIM peripheral to generate four different signals at four different frequencies.	MX
		TIM_OnePulse	This example shows how to use the TIM peripheral to generate a single pulse when a rising edge of an external signal is received on the TIMER input pin.	MX
		TIM_PWMInput	This example describes 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 timebase of one second with the corresponding interrupt request.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples	UART	LPUART_WakeUpFromStop	This example describes how to configure an LPUART to wake up the MCU from Stop mode when a given stimulus is received.	MX
		UART_HyperTerminal_DMA	This example describes UART transmission (transmit/receive) in DMA mode between a board and an HyperTerminal PC application.	MX
		UART_HyperTerminal_IT	This example describes UART transmission (transmit/receive) in Interrupt mode between a board and an HyperTerminal PC application.	MX
		UART_Printf	This example describes how to re-route the C library printf function to the UART.	MX
		UART_ReceptionToIdle_CircularDMA	This example describes how to use the HAL UART API for reception to IDLE event in circular DMA mode	MX
		UART_TwoBoards_ComDMA	This example describes UART transmission (transmit/receive) in DMA mode between two boards.	MX
		UART_TwoBoards_ComIT	This example describes UART transmission (transmit/receive) in Interrupt mode between two boards.	MX
		UART_TwoBoards_ComPolling	This example describes UART transmission (transmit/receive) in Polling mode between two boards.	MX
		UART_WakeUpFromStopUsingFIFO	This example describes how to configure an UART to wake up the MCU from Stop mode with a FIFO level when a given stimulus is received.	MX
	USART	USART_SlaveMode	This example describes USART-SPI communications (transmit/receive) between two boards where the USART is configured as a slave.	MX
	WWDG	WWDG_Example	This example describes 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
	Total number of examples: 116			**116**
Examples_LL	ADC	ADC_AnalogWatchdog_Init	This example describes 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	This example describes how to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	MX
		ADC_ContinuousConversion_TriggerSW_LowPower_Init	This example describes how to use an ADC peripheral with ADC low-power features.	MX
		ADC_Oversampling_Init	This example describes how to use an ADC peripheral with ADC oversampling.	MX
		ADC_SingleConversion_TriggerSW_DMA_Init	This example describes 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).	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples_LL	ADC	ADC_SingleConversion_TriggerSW_IT_Init	This example describes 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, refer to other examples).	MX
		ADC_SingleConversion_TriggerSW_Init	This example describes 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, refer to other examples).	MX
		ADC_SingleConversion_TriggerTimer_DMA_Init	This example describes how to use an ADC peripheral to perform a single ADC conversion on a channel at each trigger event from a timer. Converted data is indefinitely transferred by DMA into a table (Circular mode).	MX
		ADC_TemperatureSensor_Init	This example describes how to use an ADC peripheral to perform a single ADC conversion on the internal temperature sensor and calculate the temperature in Celsius degrees.	MX
	COMP	COMP_CompareGpioVsVrefInt_IT_Init	This example describes 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 STM32WLxx COMP LL API. The peripheral initialization uses the LL initialization function to demonstrate LL initialization usage.	MX
		COMP_CompareGpioVsVrefInt_OutputGpio_Init	This example describes 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 STM32WLxx COMP LL API.	MX
		COMP_CompareGpioVsVrefInt_Window_IT_Init	This example describes 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 STM32WLxx 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. This example configures a memory area as privileged read-only, and attempts to perform read and write operations in different modes.	MX
	CRC	CRC_CalculateAndCheck	This example shows 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
		CRC_UserDefinedPolynomial	This example shows 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
	DAC	DAC_GenerateConstantSignal_TriggerSW_Init	This example describes how to use the DAC peripheral to generate a constant voltage signal. This example is based on the STM32WLxx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		DAC_GenerateConstantSignal_TriggerSW_LP_Init	This example describes how to use the DAC peripheral to generate a constant voltage signal using the DAC sample-and-hold low-power feature. To be effective, a capacitor must be connected to the DAC channel output and the sample-and-hold timings must be tuned depending on the capacitor value. This example is based on the STM32WLxx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		DAC_GenerateWaveform_TriggerHW_Init	This example describes how to use the DAC peripheral to generate a voltage waveform from a digital data stream transferred by DMA. This example is based on the STM32WLxx DAC LL API. The peripheral initialization uses LL initialization functions to demonstrate LL initialization usage.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples_LL	DMA	DMA_CopyFromFlashToMemory_Init	This example describes 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
	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 STM32WLxx LL API. The peripheral initialization is done using LL initialization function to demonstrate LL initialization usage.	MX
	GPIO	GPIO_InfiniteLedToggling_Init	This example shows how to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32WLxx LL API. The peripheral is initialized with LL initialization function to demonstrate LL initialization usage.	MX
	HSEM	HSEM_DualProcess	This example describes 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	This example describes 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_AdvCommunication_DMAAndIT_Init	This example shows 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.	MX
		I2C_OneBoard_Communication_DMAAndIT_Init	This example shows 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	This example shows 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	This example shows 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	This example shows 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	This example shows 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	This example shows 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_WakeUpFromStop2_IT_Init	This example shows how to handle the reception of a data byte from an I2C slave device in Stop2 mode by an I2C master device, both using Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	MX
		I2C_TwoBoards_WakeUpFromStop_IT_Init	This example shows how to handle the reception of a data byte from an I2C slave device in Stop1 mode by an I2C master device, both using Interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	MX
	IPCC	IPCC_SimplexMode_DualCore	This example shows how to transfer data between two processors using IPCC Simplex channel mode.	MX
	IWDG	IWDG_RefreshUntilUserEvent_Init	This example shows how to configure the IWDG peripheral to ensure periodical counter update and generate an MCU IWDG reset when a user-pushbutton (B1) is pressed. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples_LL	LPTIM	LPTIM_PulseCounter_Init	This example describes 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 STM32WLxx LPTIM LL API. The peripheral is initialized with LL initialization function to demonstrate LL initialization usage.	MX
	LPUART	LPUART_WakeUpFromStop2_Init	This example describes how to configure GPIO and LPUART peripherals to enable 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	This example describes how to configure GPIO and LPUART peripherals to enable 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
	PKA	PKA_ECDSA_Sign	This example describes how to use the low-layer PKA API to generate an ECDSA signature.	MX
		PKA_ModularExponentiation	This example describes how to use the low-layer PKA API to execute RSA modular exponentiation.	MX
	PWR	PWR_EnterStandbyMode	This example shows how to enter Standby mode and wake up from this mode by using an external reset or a wakeup pin.	MX
		PWR_EnterStopMode	This example shows how to enter Stop2 mode.	MX
		PWR_OptimizedRunMode	This example shows how to increase/decrease frequency and V_{CORE} and enter/exit Low-power run mode.	MX
	RCC	RCC_HWAutoMSICalibration	This example shows how to use of the MSI clock source hardware autocalibration and LSE clock (PLL mode) to obtain an accurate MSI clock.	MX
		RCC_OutputSystemClockOnMCO	This example shows how to configure MCO pin (PA8) to output the system clock.	MX
		RCC_UseHSEasSystemClock	This example shows how to use the RCC LL API to start the HSE and use it as system clock.	MX
		RCC_UseHSI_PLLasSystemClock	This example shows how to modify the PLL parameters in runtime.	MX
	RNG	RNG_GenerateRandomNumbers	This example shows 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	This example shows 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
	RTC	RTC_Alarm_Init	This example shows 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_ExitStandbyWithWakeUpTimer_Init	This example shows how to periodically enter and wake up from Standby mode thanks to the RTC Wakeup Timer (WUT).	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples_LL	RTC	RTC_Tamper_Init	This example shows 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	This example shows 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_Init	This example shows 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 STM32WLxx SPI LL API. The peripheral initialization uses the LL initialization function to demonstrate LL initialization usage.	MX
		SPI_OneBoard_HalfDuplex_IT_Init	This example shows how to configure to 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 STM32WLxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		SPI_TwoBoards_FullDuplex_DMA_Master_Init	This example shows how to perform data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32WLxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		SPI_TwoBoards_FullDuplex_DMA_Slave_Init	This example shows how to perform data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32WLxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		SPI_TwoBoards_FullDuplex_IT_Master_Init	This example shows how to perform data buffer transmission and reception via SPI using Interrupt mode. This example is based on the STM32WLxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		SPI_TwoBoards_FullDuplex_IT_Slave_Init	This example shows how to perform data buffer transmission and reception via SPI using Interrupt mode. This example is based on the STM32WLxx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
	TIM	TIM_BreakAndDeadtime_Init	This example shows 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.	MX
		TIM_DMA_Init	This example shows 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 STM32WLxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		TIM_InputCapture_Init	This example shows 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 STM32WLxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		TIM_OnePulse_Init	This example shows 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 STM32WLxx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL initialization.	MX
		TIM_OutputCompare_Init	This example shows how to configure the TIM peripheral to generate an output waveform in different output compare modes. This example is based on the STM32WLxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		TIM_PWMOutput_Init	This example shows how to use a TIM peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32WLxx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL initialization.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples_LL	TIM	TIM_TimeBase_Init	This example shows how to configure the TIM peripheral to generate a timebase. This example is based on the STM32WLxx 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 GPIO and USART peripheral to continuously receive characters from an 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_Continuous_VCP_Init	This example shows how to configure GPIO and USART peripheral to continuously receive characters from an 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	This example shows how to configure GPIO and USART peripheral to receive characters from an 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_Rx_IT_VCP_Init	This example shows how to configure GPIO and USART peripheral to receive characters from an 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	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to/from an HyperTerminal (PC) in DMA mode. This example is based on STM32WLxx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX
		USART_Communication_Tx_IT_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to an HyperTerminal (PC) in Interrupt mode. This example is based on STM32WLxx 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	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to an HyperTerminal (PC) in Interrupt mode. This example is based on STM32WLxx USART LL API. The 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 an HyperTerminal (PC) in Polling mode. If the transfer cannot be complete within the allocated time, a timeout allows to exit from the sequence with a Timeout error code. This example is based on STM32WLxx USART LL API. The 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 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 STM32WLxx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX
		USART_HardwareFlowControl_Init	This example shows how to configure GPIO and USART peripheral to receive characters asynchronously from an HyperTerminal (PC) in Interrupt mode with the Hardware flow control feature enabled. This example is based on STM32WLxx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		USART_SyncCommunication_FullDuplex_DMA_Init	This example shows how to configure GPIO, USART, DMA and SPI peripherals to transmit bytes between a USART and an SPI (in Slave mode) in DMA mode. This example is based on the STM32WLxx 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-WL55JC
Examples_LL	USART	USART_SyncCommunication_FullDuplex_IT_Init	This example shows how to configure GPIO, USART, DMA and SPI peripherals to transmit bytes between a USART and an SPI (in Slave mode) in Interrupt mode. This example is based on the STM32WLxx USART LL API (the SPI uses the DMA to receive/transmit characters sent from/received by the USART). The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX
		USART_WakeUpFromStop1_Init	This example shows how to configure GPIO and USART1 peripherals to allow the characters received on USART_RX pin to wake up the MCU from low-power mode.	MX
		USART_WakeUpFromStop_Init	This example shows how to configure GPIO and USART1 peripherals to allow the characters received on USART_RX pin to wake up the MCU from low-power mode.	MX
	UTILS	UTILS_ConfigureSystemClock	This example shows how to use UTILS LL API to configure the system clock using PLL with HSI as source clock.	MX
		UTILS_ReadDeviceInfo	This example reads the UID, Device ID and Revision ID and saves them into a global information buffer.	MX
	WWDG	WWDG_RefreshUntilUserEvent_Init	This example shows 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
	Total number of examples_ll: 82			**82**
Examples_MIX	ADC	ADC_SingleConversion_TriggerSW_IT	This example describes 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 STM32WLxx ADC HAL and LL API. The LL API is used for performance improvement.	MX
	CRC	CRC_PolynomialUpdate	This example describes how to use the CRC peripheral through the STM32WLxx CRC HAL and LL API.	MX
	DMA	DMA_FLASHToRAM	This example describes how to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the STM32WLxx DMA HAL and LL API. The LL API is used for performance improvement.	MX
	I2C	I2C_OneBoard_ComSlave7_10bits_IT	This example shows 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 STM32WLxx I2C HAL and LL API (LL API usage for performance improvement) and an interrupt.	MX
	PWR	PWR_STOP1	This example shows how to enter Stop1 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
	SPI	SPI_FullDuplex_ComPolling_Master	This example shows how to perform data buffer transmission/reception between two boards via SPI using Polling mode.	MX
		SPI_FullDuplex_ComPolling_Slave	This example shows how to perform data buffer transmission/reception between two boards via SPI using Polling mode.	MX
		SPI_HalfDuplex_ComPollingIT_Master	This example shows how to perform data buffer transmission/reception between two boards via SPI using Polling (LL driver) and Interrupt modes (HAL driver).	MX
		SPI_HalfDuplex_ComPollingIT_Slave	This example shows how to perform data buffer transmission/reception between two boards via SPI using Polling (LL driver) and Interrupt modes (HAL driver).	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Examples_MIX	TIM	TIM_PWMInput	This example shows how to use the TIM peripheral to measure an external signal frequency and duty cycle.	MX
	UART	UART_HyperTerminal_IT	This example shows how to use a UART to transmit data (transmit/receive) between a board and an HyperTerminal PC application in Interrupt mode. This example describes how to use the USART peripheral through the STM32WLxx UART HAL and LL API, the LL API being used for performance improvement.	MX
		UART_HyperTerminal_TxPolling_RxIT	This example shows how to use a UART to transmit data (transmit/receive) 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 STM32WLxx UART HAL and LL API, the LL API being used for performance improvement.	MX
	Total number of examples_mix: 12			**12**
Applications	BFU_1_Image	1_Image_BFU	The boot (B) and firmware update (FU) solution enables the update of the STM32 microcontroller built-in program with new firmware versions to add new features and correct potential issues. The update process is performed in a secure way to prevent unauthorized updates.	X
		1_Image_KMS_Blob	This application is used to generate a key management service blob binary file to be downloaded with KMS through the ImportBlob() API.	X
		1_Image_SECoreBin	This application is used to generate a secure engine core binary file to be linked with the boot and firmware update application (BFU).	X
		1_Image_UserApp	This application demonstrates firmware download capabilities and provides a set of functions to test the active protections offered by the secure boot and secure engine.	X
	FatFs	FatFs_uSD_Standalone	This application shows how to use the STM32Cube MCU Package 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.	X
	FreeRTOS	FreeRTOS_HSEM_DualCore	This example shows how to synchronise two CPUs using HSEM peripherals and CMSISOS2.0 RTOS API application.	X
		FreeRTOS_LowPower	This example shows how to enter and exit low-power mode with CMSIS RTOS API.	MX
		FreeRTOS_Queues	This application shows how to use message queues with CMSIS RTOS API.	MX
		FreeRTOS_Semaphore	This application shows how to use semaphores with CMSIS RTOS API.	MX
		FreeRTOS_StopMode	This example shows how to enter and exit low-power mode with CMSISOS2.0 RTOS API.	MX
		FreeRTOS_ThreadCreation	This example shows how to implement thread creation using CMSIS RTOS API.	MX
		FreeRTOS_ThreadFlags	This application shows how to use thread flags with CMSIS_RTOS2 API.	MX

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Applications	FreeRTOS	FreeRTOS_Timers	This application shows how to use timers of CMSIS RTOS API.	MX
	KMS	KMS_Blob_Binary	This application is used to generate key management services (KMS) blob binary file to be downloaded with KMS through the ImportBlob() API.	X
		KMS_Blob_Example	This application shows how to use the KMS middleware provisioning capabilities (C_STM_ImportBlob API).	MX
		KMS_Derive_Key	This application shows how to use the KMS middleware to create a derivative AES key.	MX
		KMS_Embedded_AES_Keys	This application shows how to use the KMS middleware to encrypt and decrypt message with an AES key.	MX
		KMS_Embedded_RSA_Key	This application shows how to use the KMS middleware to sign and verify a message with an RSA key.	MX
	LoRaWAN	LoRaWAN_AT_Slave	This directory contains a set of source files that implements a LoRa® application modem that is controlled by an external host though an AT command interface over UART, like a computer executing a terminal application.	MX
		LoRaWAN_AT_Slave_DualCore	This directory contains a set of source files that implements a dual-core (Arm® Cortex®-M4/-M0) LoRa® application modem that is controlled by an external host though an AT command interface over UART, like a computer executing a terminal application.	MX
		LoRaWAN_End_Node	This directory contains a set of source files that implements a LoRa® application device sending sensor data to a LoRa® network server.	MX
		LoRaWAN_End_Node_DualCore	This directory contains a set of source files that implements a dual-core (Arm® Cortex®-M4/ -M0) LoRa® application device sending sensor data to a LoRa® network server.	MX
		LoRaWAN_End_Node_DualCoreFreeRTOS	This directory contains a set of source files that implements a dual-core (Arm® Cortex®-M4/-M0) LoRa® application device sending sensor data to a LoRa® network server.	MX
	LoRaWAN_FUOTA	2_Images_KMS_Blob	This application is used to generate KMS blob binary files to be downloaded with KMS through the ImportBlob() API.	X
		2_Images_SBSFU	The secure boot (SB) and secure firmware update (SFU) solution enables updating the STM32 microcontroller built-in programs with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way to prevent unauthorized updates and accesses to confidential on-device data such as secret code and firmware encryption key.	X
		2_Images_SECoreBin	This application is used to generate secure engine core binary file to be linked with the secure boot and secure firmware update application (SBSFU).	X
		LoRaWAN_End_Node_DualCore	This directory contains a set of source files that implements a dual-core (Arm® Cortex®-M4/-M0) LoRa® application device that sending sensor data a to LoRa® network server.	X
	SBSFU_2_Images _DualCore	2_Images_KMS_Blob	This application is used to generate KMS blob binary file to be downloaded with KMS through the ImportBlob() API.	X

Level	Module Name	Project Name	Description	NUCLEO-WL55JC
Applications	SBSFU_2_Images_DualCore	2_Images_SBSFU	The secure boot (SB) and secure firmware update (SFU) solution enables updating the STM32 microcontroller built-in program with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way to prevent unauthorized updates and accesses to confidential on-device data such as secret code and firmware encryption key.	X
		2_Images_SECoreBin	This application is used to generate secure engine core binary file to be linked with secure boot and secure firmware update application (SBSFU).	X
		2_Images_UserApp_M0Plus	This application demonstrates firmware download capabilities and provides a set of functions to test the active protections offered by secure boot and secure engine.	X
		2_Images_UserApp_M4	This application demonstrates firmware download capabilities.	X
	Sigfox	SigFox_AT_Slave	This directory contains a set of source files that implements a Sigfox™ application modem that is controlled though AT command interface over UART by an external host, like a computer executing a terminal.	MX
		SigFox_AT_Slave_DualCore	This directory contains a set of source files that implements a dual-core (Arm® Cortex®-M4/-M0) Sigfox™ application modem that is controlled though AT command interface over UART by an external host, like a computer executing a terminal.	MX
		SigFox_PushButton	This directory contains a set of source files that implements an example of a Sigfox™ object sending temperature - and battery-level to a Sigfox™ network.	MX
		SigFox_PushButton_DualCore	This directory contains a set of source files that implements a dual-core (Arm® Cortex®-M4 /-M0) Sigfox™ application example of a Sigfox™ object sending temperature and battery level to a Sigfox™ network.	MX
	SubGHz_Phy	SubGHz_Phy_PingPong	This directory contains a set of source files that implements a ping-pong application between two PingPong devices.	MX
		SubGHz_Phy_PingPong_DualCore	This directory contains a set of source files that implements a dual core (Arm® Cortex®-M4 /-M0) ping-pong application between two PingPong devices.	MX
	Total number of applications: 38			**38**
Demonstrations	Local Network	LocalNetwork_Concentrator	This directory contains a set of source files that implements a concentrator that sends beacons to administrate a network of up to 14 sensors and receives data from each connected sensor. This application targets the NUCLEO-WL55JC.	X
		LocalNetwork_Sensor	This directory contains a set of source files that implements a sensor that sends sensor data to the demonstration concentrator. This application targets the NUCLEO-WL55JC board.	X
	Total number of demonstrations: 2			**2**
Total number of projects: 254				**254**

Revision history

Table 2. **Document revision history**

Date	Version	Changes
20-Oct-2020	1	Initial release.

Contents