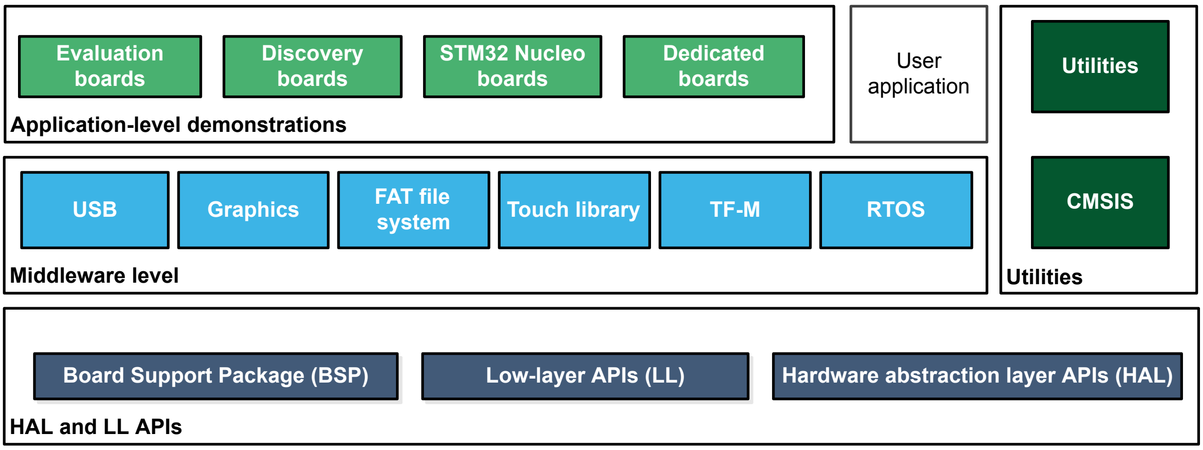
STM32Cube firmware examples for STM32L5 Series

Introduction

The STM32CubeL5 MCU Package is delivered with a rich set of examples running on STMicroelectronics boards. The examples are organized by boards and provided with pre-configured projects for the main supported toolchains (Refer to Figure 1).

Figure 1. STM32CubeL5 firmware components



1 Reference documents

The following items make up a reference set for the examples presented in this application note:

- The latest release of the STM32CubeL5 MCU Package for the 32-bit microcontrollers in the STM32L5 Series based on the Arm® Cortex®-M processor with Arm®TrustZone®
- *Getting started with STM32CubeL5 for STM32L5 Series* (UM2656)
- *Description of STM32L5 HAL and Low Layer Drivers* (UM2659)
- *STM32Cube USB device library* (UM1734)
- *Developing Applications on STM32Cube with FatFS* (UM1721)
- *Developing applications on STM32Cube with RTOS* (UM1722)
- *Getting started with STM32CubeL5 TFM application* (UM2671)
- *Overview of Secure Boot and Secure Firmware Update solution on Arm® TrustZone® STM32L5 Series microcontrollers* (AN5447)

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

2 STM32CubeL5 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 not used). Their objective is to demonstrate the product or peripheral features and usage. They are organized per peripheral (One folder per peripheral, such as TIM). Their complexity level ranges from the basic usage of a given peripheral, such as PWM generation using a timer, to the integration of several peripherals, such as how to use DAC for a 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 drivers and middleware components not used). They offer an optimum implementation of typical use cases of the peripheral features and configuration sequences. The examples are organized per peripheral (One folder for each peripheral, such as TIM) and are principally deployed on Nucleo boards.
- **Examples_MIX**

  These examples use only 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 offers high-level function-oriented APIs with high portability level by hiding product/IPs complexity for end-users.
  - LL provides low-level APIs at the register level with better optimization.

  The examples are organized per peripheral (One folder for each peripheral, such as TIM) and are exclusively deployed on Nucleo boards.
- **Applications**

  The applications demonstrate product performance and how to use the available middleware stacks. They are organized either by middleware (One folder per middleware, such as USB host) or product feature that requires high-level firmware bricks (Such as 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.
- **Template_LL project**

  The template LL projects are provided to allow the user to quickly build a firmware application using LL drivers on a given board.

The examples are located under `STM32Cube_FW_L5_VX.Y.Z\Projects\`.

The examples in the default product configuration with the Arm® TrustZone® disabled have the same structure:

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

The examples with the Arm® TrustZone® enabled are suffixed with "_TrustZone" (except TFM applications) and have the same structure:

- `*\Secure\Inc` folder, containing all secure project header files
- `*\Secure\Src` and `*\Secure_nsclib\` folders, containing all secure project sources code
- `*\NonSecure\Inc` folder, containing all non-secure project header files
- `*\Non\Secure\Src` folder, containing all non-secure project sources code
- `*\EWARM`, `*\MDK-ARM`, and `*\STM32CubeIDE` folders, containing the preconfigured project for each toolchain
- `*\readme.txt` file, describing the example behavior and the environment required to run the example

To run the example, proceed as follows:

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

Note: *Refer to "Development toolchains and compilers" and "Supported devices and evaluation boards" sections of the firmware package release notes to know more about the software/hardware environment used for the MCU Package development and validation. The correct operation of the provided examples is not guaranteed in other environments, for example, when using different compilers 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, pushbuttons, and others). The BSP is based on a modular architecture that can be easily ported to any hardware by implementing low-level routines.

Table 1 contains the list of examples provided with the STM32CubeL5 MCU Package.

In this table, the label MX means the projects are created using STM32CubeMX, the STM32Cube initialization code generator. Those projects can be opened with this tool to modify the projects themselves. The other projects are manually created to demonstrate the product features. In this table, the label TrustZone means the projects are created for devices with Arm® TrustZone® enabled. Read the project `readme.txt` file for user option bytes configuration.

Table 1. **STM32CubeL5 firmware examples**

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Templates	-	TrustZoneDisabled	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application when TrustZone security is not enabled (TZEN=0).	X	X	X
		TrustZoneEnabled	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application when TrustZone security is activated (Option bit TZEN=1).	X TrustZone	X TrustZone	X TrustZone
	Total number of templates: 6			**2**	**2**	**2**
Templates_LL	-	TrustZoneDisabled	Reference template based on the STM32Cube LL API that can be used to build any firmware application.	X	X	X
	Total number of templates_ll: 3			**1**	**1**	**1**
Examples	-	BSP	How to use the different BSP drivers of the board.	X	X	-
	ADC	ADC_AnalogWatchdog	How to use the ADC peripheral to perform conversions with an analog watchdog and out-of-window interrupts enabled.	-	-	MX
		ADC_MultiChannelSingleConversion	Use ADC to convert several channels using sequencer in discontinuous mode, conversion data are transferred by DMA into an array, indefinitely (Circular mode).	-	MX	MX
		ADC_Oversampling	Use ADC to convert a single channel but using the oversampling feature to increase resolution.	-	MX	MX
		ADC_SingleConversion_TriggerSW_IT	Use ADC to convert a single channel at each SW start, conversion performed using programming model: interrupt Example configuration: ADC is configured to convert a single channel, in single conversion mode, from the software trigger.	-	-	MX
		ADC_SingleConversion_TriggerTimer_DMA	Use ADC to convert a single channel at each trig from the timer, conversion data are transferred by DMA into an array, indefinitely (Circular mode).	-	-	MX
	COMP	COMP_CompareGpioVsVrefInt_IT	How to configure the COMP peripheral to compare the external voltage applied on a specific pin with the Internal Voltage Reference.	MX	MX	MX
		COMP_CompareGpioVsVrefInt_Window_IT	This example shows how to make an analog watchdog using the COMP peripherals in window mode.	MX	MX	MX
	CORTEX	CORTEXM_InterruptSwitch_TrustZone	How to first use an interrupt in the secure application and later assign it to the non-secure application when TrustZone security is activated (Option bit TZEN=1).	-	-	MX TrustZone

Level	Module Name	Project Name	Description	STM32L562E-DK(1)	STM32L552E-EV(1)	NUCLEO-L552ZE-Q(1)
Examples	CORTEX	CORTEXM_ModePrivilege	How to modify the Thread mode privilege access and stack. Thread mode is entered on reset or when returning from an exception.	MX	-	MX
		CORTEXM_ProcessStack	How to modify the Thread mode stack. Thread mode is entered on reset and can be entered as a result of an exception return.	MX	-	MX
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms time base to toggle LEDs.	MX	-	MX
	CRC	CRC_Bytes_Stream_7bit_CRC	How to configure the CRC using the HAL API. The CRC (Cyclic redundancy check) calculation unit computes 7-bit CRC codes derived from buffers of 8-bit data (Bytes). The user-defined generating polynomial is manually set to 0x65, that is, X^7 + X^6 + X^5 + X^2 + 1, as used in the Train Communication Network, IEC 60870-5[17].	MX	-	MX
		CRC_Data_Reversing_16bit_CRC	How to configure the CRC using the HAL API. The CRC (Cyclic redundancy check) calculation unit computes a 16-bit CRC code derived from a buffer of 32-bit data (Words). Input and output data reversal features are enabled. The user-defined generating polynomial is manually set to 0x1021, that is, X^16 + X^12 + X^5 + 1, which is the CRC-CCITT generating polynomial.	MX	-	MX
		CRC_Example	How to configure the CRC using the HAL API. The CRC (Cyclic redundancy check) calculation unit computes the CRC code of a given buffer of 32-bit data words, using a fixed generator polynomial (0x4C11DB7).	MX	-	MX
		CRC_UserDefinedPolynomial	How to configure the CRC using the HAL API. The CRC (Cyclic redundancy check) calculation unit computes the 8-bit CRC code for a given buffer of 32-bit data words, based on a user-defined generating polynomial.	MX	-	MX
	CRYP	CRYP_AESModes	How to use the CRYP peripheral to encrypt and decrypt data using AES in chaining modes (ECB, CBC, CTR).	MX	-	-
		CRYP_AESModes_Suspension	How to use the CRYP peripheral to suspend then resume ciphering processing.	MX	-	-
		CRYP_DMA	How to use the CRYP peripheral to encrypt and decrypt data using the AES-128 algorithm with ECB chaining mode in DMA mode.	MX	-	-
		CRYP_GCM_GMAC_CCM_Modes	How to use the CRYP peripheral to encrypt data and generate authentication tags using GCM/GMAC/CCM modes.	MX	-	-
		CRYP_GCM_Suspension	How to use the CRYP peripheral to suspend then resume an authentication ciphering processing.	MX	-	-
	DAC	DAC_SimpleConversion	How to use the DAC peripheral to do a simple conversion.	-	MX	MX

Level	Module Name	Project Name	Description	STM32L562E-DK(1)	STM32L552E-EV(1)	NUCLEO-L552ZE-Q(1)
Examples	DFSDM	DFSDM_AudioRecord	How to use the DFSDM HAL API to perform mono audio recording. This example uses the SPH0641LM4H-1 digital microphone mounted on the board.	MX	MX	-
		DFSDM_Thermometer	How to use the DFSDM HAL API to perform temperature measurements. This example uses the PTS100R (Thermistor) and STPMS2 (Sigma-delta modulator) mounted on the board. The STPMS2 allows voltage and current values to be obtained from the PTS100R. The temperature value is thus deduced.	-	MX	-
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the HAL API.	MX	MX	MX
		DMA_MUXSYNC	How to use the DMA with the DMAMUX to synchronize a transfer with the LPTIM1 output signal. USART3 is used in DMA synchronized mode to send a countdown from 10 to 00, with a period of 2 seconds.	-	-	MX
		DMA_MUX_RequestGen	How to use the DMA with the DMAMUX request generator to generate DMA transfer requests upon an External line 13 rising edge signal.	-	-	MX
		DMA_MemToMem_TrustZone	How to use HAL DMA to perform memory to memory data transfers over secure and non-secure DMA channels when TrustZone security is activated (Option bit TZEN=1).	-	-	MX TrustZone
	FDCAN	FDCAN_Classic_Frame_Networking	How to configure the FDCAN peripheral to send and receive Classic CAN frames.	-	MX	-
		FDCAN_Loopback	How to configure the FDCAN to operate in loopback mode.	-	MX	-
	FLASH	FLASH_BlockBased_TrustZone	How to configure and use the FLASH HAL API to managed block-based security of internal Flash memory between secure and non-secure applications when TrustZone security is activated (Option bit TZEN=1).	-	MX TrustZone	-
		FLASH_DualBoot	Guide through the configuration steps to program internal Flash memory bank 1 and bank 2, and to swap between both of them using the FLASH HAL API.	MX	MX	-
		FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal Flash memory.	MX	MX	MX
		FLASH_EraseProgram_TrustZone	How to configure and use the FLASH HAL API to erase and program the internal Flash memory when TrustZone security is activated (Option bit TZEN=1).	-	MX TrustZone	-
		FLASH_WriteProtection	How to configure and use the FLASH HAL API to enable and disable the write protection of the internal Flash memory.	MX	MX	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[(1)]	STM32L552E-EV[(1)]	NUCLEO-L552ZE-Q[(1)]
Examples	FMC	FMC_SRAM	How to configure the FMC controller to access the IS61WV51216BLL SRAM memory.	-	MX	-
		FMC_SRAM_DataMemory	How to configure the FMC controller to access the IS61WV51216BLL SRAM memory including heap and stack.	-	MX	-
		FMC_SRAM_TrustZone	How to configure the FMC controller to access the IS61WV51216BLL SRAM memory split between secure and non-secure applications when TrustZone security is activated (Option bit TZEN=1). .	-	MX TrustZone	-
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	MX	-	MX
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	MX	MX	MX
		GPIO_IOToggle_TrustZone	How to use HAL GPIO to toggle secure and unsecured IOs when TrustZone security is activated (Option bit TZEN=1).	MX TrustZone	-	MX TrustZone
	GTZC	GTZC_MPCWM_IllegalAccess_TrustZone	How to use GTZC MPCWM-TZIC to build any example when TrustZone security is activated (Option bit TZEN=1).	-	MX TrustZone	-
		GTZC_TZSC_MPCBB_TrustZone	How to use HAL GTZC MPCBB to build any example with SecureFault detection when TrustZone security is activated (Option bit TZEN=1).	MX TrustZone	-	MX TrustZone
	HAL	HAL_RegisterCallbacks_TIM	Register a callback function called every second based on TIM peripheral configuration to generate a time base of one second with the corresponding interrupt request.	-	-	X
		HAL_TimeBase_RTC_WKUP	How to customize HAL using RTC wake-up as the main source of the time base, instead of Systick.	-	-	MX
		HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as the main source of the time base instead of Systick.	MX	MX	MX
	HASH	HASH_HMAC_SHA1MD5	How to use the HASH peripheral to hash data with HMAC SHA-1 and HMAC MD5 algorithms.	-	-	MX
		HASH_HMAC_SHA224SHA1_DMA_Suspension	How to suspend the HMAC digest computation when data are fed to the HASH unit with DMA.	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples	HASH	HASH_HMAC_SHA224SHA256_MultiBuffer_DMA	How to handle text messages larger than the maximum DMA transfer length. The input data are split into several buffers with sizes within the DMA limit, then fed successively to the HASH peripheral.	-	-	MX
		HASH_HMAC_SHA256MD5_IT_Suspension	How to suspend the HMAC digest computation when data are fed in interrupt mode.	-	-	MX
		HASH_SHA1MD5	This example shows how to use the HASH peripheral to hash data with SHA-1 and MD5 algorithms.	-	-	MX
		HASH_SHA1MD5_DMA	How to use the HASH peripheral to hash data using SHA-1 and MD5 algorithms when data are fed to the HASH unit with DMA.	-	-	MX
		HASH_SHA1SHA224_IT_Suspension	How to suspend the HASH peripheral when data are fed in interrupt mode.	-	-	MX
		HASH_SHA1_DMA_TrustZone	How to use a secure HASH SHA-1 computation service based on a secure DMA channel when TrustZone security is activated (Option bit TZEN=1).	-	-	MX TrustZone
		HASH_SHA224SHA256_DMA	How to use the HASH peripheral to hash data with SHA224 and SHA256 algorithms.	-	-	MX
		HASH_SHA256MD5_DMA_Suspension	How to suspend the HASH peripheral when data are fed to the HASH unit with DMA.	-	-	MX
	I^2C	I2C_TwoBoards_AdvComIT	How to handle I^2C data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
		I2C_TwoBoards_ComDMA	How to handle I^2C data buffer transmission/reception between two boards, via DMA.	-	-	MX
		I2C_TwoBoards_ComIT	How to handle I^2C data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
		I2C_TwoBoards_ComPolling	How to handle I^2C data buffer transmission/reception between two boards, in polling mode.	-	-	MX
		I2C_TwoBoards_RestartAdvComIT	How to perform multiple I^2C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	-	-	MX
		I2C_TwoBoards_RestartComIT	How to handle single I^2C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK(1)	STM32L552E-EV(1)	NUCLEO-L552ZE-Q(1)
Examples	I2C	I2C_WakeUpFromStop2	How to handle I2C data buffer transmission/reception between two boards, using an interrupt when the device is in Stop 2 mode.	-	-	MX
	ICACHE	ICACHE_SRAM_Memory_Remap	How to execute code from an external SRAM remapped region configured through the ICACHE HAL driver.	-	MX	-
	IWDG	IWDG_Reset	How to handle the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset lap of time.	-	MX	MX
		IWDG_WindowMode	How to periodically update the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset lap of time.	-	-	MX
	LPTIM	LPTIM_PWMExternalClock	How to configure and use, through the HAL LPTIM API, the LPTIM peripheral using an external counter clock, to generate a PWM signal at the lowest power consumption.	-	MX	-
		LPTIM_PWM_LSE	How to configure and use, through the HAL LPTIM API, the LPTIM peripheral using LSE as a counter clock, to generate a PWM signal, in a low-power mode.	-	MX	-
		LPTIM_PulseCounter	How to configure and use, through the LPTIM HAL API, the LPTIM peripheral to count pulses.	MX	MX	MX
		LPTIM_Timeout	How to implement, through the HAL LPTIM API, a timeout with the LPTIMER peripheral, to wake up the system from a low-power mode.	-	MX	MX
	OCTOSPI	OSPI_NOR_ExecuteInPlace	How to execute code from an OSPI memory after code loading.	MX	MX	-
		OSPI_NOR_ExecuteInPlace_DTR	How to execute code from an OSPI memory after code loading.	MX	MX	-
		OSPI_NOR_MemoryMapped	How to use an OSPI NOR memory in memory-mapped mode.	MX	MX	-
		OSPI_NOR_MemoryMapped_DTR	How to use an OSPI NOR memory in memory-mapped mode.	MX	MX	-
		OSPI_NOR_ReadWrite_DMA	How to use an OSPI NOR memory in DMA mode.	MX	MX	-
		OSPI_NOR_ReadWrite_DMA_DTR	How to use an OSPI NOR memory in DMA mode.	MX	MX	-
		OSPI_RAM_ExecuteInPlace	How to execute code from an OSPI memory after code loading.	-	MX	-

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples	OCTOSPI	OSPI_RAM_MemoryMapped	How to use an OSPI HyperRAM memory in memory-mapped mode.	-	MX	-
		OSPI_RAM_ReadWrite_DMA	How to use an OSPI HyperRAM memory in DMA mode.	-	MX	-
	OPAMP	OPAMP_PGA	How to configure the OPAMP peripheral in PGA mode (OPAMP programmable gain).	-	-	MX
		OPAMP_STANDALONE	How to configure the OPAMP peripheral in standalone mode. The gain in this mode can be set externally (External gain setting mode).	-	-	MX
	OTFDEC	OTFDEC_Ciphering_TrustZone	How to use a secure OTFDEC (On-The-Fly Decoder EnCoder) when TrustZone security is activated (Option bit TZEN=1) to cipher data from the secure side and to allow to decipher from non-secure without any key exchange.	MX TrustZone	-	-
		OTFDEC_DataDecrypt	How to decrypt data located on the OCTOSPI external flash using the OTFDEC peripheral.	MX	-	-
		OTFDEC_ExecutingCryptedInstruction	How to execute ciphered instructions stored in external NOR flash using the OTFDEC peripheral.	MX	-	-
	PKA	PKA_ECCscalarMultiplication	How to use the PKA peripheral to execute ECC scalar multiplication. This allows generating a public key from a private key.	MX	-	-
		PKA_ECCscalarMultiplication_IT	How to use the PKA peripheral to execute ECC scalar multiplication. This allows generating a public key from a private key in interrupt mode.	MX	-	-
		PKA_ECDSA_Sign	How to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA).	MX	-	-
		PKA_ECDSA_Sign_IT	How to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA) in interrupt mode.	MX	-	-
		PKA_ECDSA_Verify	How to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA).	MX	-	-
		PKA_ECDSA_Verify_IT	How to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA) in interrupt mode.	MX	-	-
		PKA_ModularExponentiation	How to use the PKA peripheral to execute modular exponentiation. This allows ciphering/deciphering a text.	MX	-	-

Level	Module Name	Project Name	Description	STM32L562E-DK(1)	STM32L552E-EV(1)	NUCLEO-L552ZE-Q(1)
Examples	PKA	PKA_ModularExponentiationCRT	How to compute the Chinese Remainder Theorem (CRT) optimization.	MX	-	-
		PKA_ModularExponentiationCRT_IT	How to compute the Chinese Remainder Theorem (CRT) optimization in interrupt mode.	MX	-	-
		PKA_ModularExponentiation_IT	How to use the PKA peripheral to execute modular exponentiation. This allows ciphering/deciphering a text in interrupt mode.	MX	-	-
		PKA_PointCheck	How to use the PKA peripheral to determine if a point is on a curve. This allows validating an external public key.	MX	-	-
		PKA_PointCheck_IT	How to use the PKA peripheral to determine if a point is on a curve. This allows validating an external public key.	MX	-	-
	PWR	PWR_LPRUN	How to enter and exit the Low-power run mode.	MX	-	MX
		PWR_LPRUN_SRAM1	How to enter and exit the Low-power run mode.	-	-	MX
		PWR_LPSLEEP	How to enter the Low-power sleep mode and wake up from this mode by using an interrupt.	MX	-	MX
		PWR_PVD	How to configure the programmable voltage detector by using an external interrupt line. External DC supply must be used to supply Vdd.	-	MX	-
		PWR_RUN_SMPS	How to use the SMPS step-down converter in RUN mode.	-	-	MX
		PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	MX	-	MX
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using an external reset or the WKUP pin.	MX	-	MX
		PWR_STOP1	How to enter the Stop 1 mode and wake up from this mode by using an interrupt.	-	-	MX
		PWR_STOP1_RTC	How to enter the Stop 1 mode and wake up from this mode by using an interrupt from the RTC wake-up timer.	-	-	MX
		PWR_STOP2	How to enter the Stop 2 mode and wake up from this mode by using an external reset or wake-up interrupt.	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples	PWR	PWR_STOP2_RTC	How to enter the Stop 2 mode and wake up from this mode by using an interrupt from the RTC wake-up timer.	-	-	MX
	RCC	RCC_CRS_Synchronization_IT	Configuration of the clock recovery service (CRS) in Interrupt mode, using the RCC HAL API.	-	-	MX
		RCC_CRS_Synchronization_Polling	Configuration of the clock recovery service (CRS) in Polling mode, using the RCC HAL API.	-	-	MX
		RCC_ClockConfig	Configuration of the system clock (SYSCLK) and modification of the clock settings in Run mode, using the RCC HAL API.	MX	MX	MX
		RCC_ClockConfig_TrustZone	Configuration of the system clock (SYSCLK) in Run mode from the secure application upon request from the non-secure application, using the RCC HAL API when TrustZone security is activated (Option bit TZEN=1).	MX TrustZone	-	MX TrustZone
		RCC_LSEConfig	Enabling/disabling of the low-speed external (LSE) RC oscillator (About 32 KHz) at run time, using the RCC HAL API.	-	-	MX
		RCC_LSIConfig	Enabling/disabling of the low-speed internal (LSI) RC oscillator (About 32 KHz) at run time, using the RCC HAL API.	-	-	MX
	RNG	RNG_MultiRNG	Configuration of the RNG using the HAL API. This example uses the RNG to generate 32-bit long random numbers.	MX	MX	MX
		RNG_MultiRNG_IT	Configuration of the RNG using the HAL API. This example uses RNG interrupts to generate 32-bit long random numbers.	-	-	MX
	RTC	RTC_ActiveTamper	Configuration of the active tamper detection with backup registers erase.	MX	MX	-
		RTC_Alarm	Configuration and generation of an RTC alarm using the RTC HAL API.	MX	MX	MX
		RTC_Calendar	Configuration of the calendar using the RTC HAL API.	-	MX	-
		RTC_LSI	Use of the LSI clock source autocalibration to get a precise RTC clock.	-	-	MX
		RTC_LowPower_STANDBY_WUT	How to periodically enter and wake up from STANDBY mode thanks to the RTC Wake-Up Timer (WUT).	MX	MX	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples	RTC	RTC_Tamper	Configuration of the tamper detection with backup registers erase.	-	-	MX
		RTC_TimeStamp	Configuration of the RTC HAL API to demonstrate the timestamp feature.	MX	-	MX
		RTC_TrustZone	How to configure the TrustZone-aware RTC peripheral when TrustZone security is activated (Option bit TZEN=1): some features of the RTC can be secure while the others are non-secure.	-	-	MX TrustZone
	SAI	SAI_AudioPlay	Use of the SAI HAL API to play an audio file in DMA circular mode and handle the buffer update.	MX	MX	-
	SPI	SPI_FullDuplex_ComDMA_Master	Data buffer transmission/reception between two boards via SPI using DMA.	-	-	MX
		SPI_FullDuplex_ComDMA_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	-	-	MX
		SPI_FullDuplex_ComIT_Master	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	-	-	MX
		SPI_FullDuplex_ComIT_Slave	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	-	-	MX
		SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	-	MX
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	-	MX
	TIM	TIM_ExtTriggerSynchro	This example shows how to synchronize TIM peripherals in cascade mode with an external trigger.	-	-	MX
		TIM_InputCapture	How to use the TIM peripheral to measure an external signal frequency.	-	-	MX
		TIM_OCActive	Configuration of the TIM peripheral in Output or Compare or Active mode (When the counter matches the capture/compare register, the corresponding output pin is set to its active state).	-	-	MX
		TIM_OCInactive	Configuration of the TIM peripheral in Output Compare Inactive mode with the corresponding Interrupt requests for each channel.	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples	TIM	TIM_OCToggle	Configuration of the TIM peripheral to generate four different signals at four different frequencies.	-	-	MX
		TIM_PWMInput	How to use the TIM peripheral to measure the frequency and duty cycle of an external signal.	-	-	MX
		TIM_PWMOutput	This example shows how to configure the TIM peripheral in PWM (Pulse Width Modulation) mode.	MX	MX	MX
	UART	LPUART_WakeUpFromStop	Configuration of an LPUART to wake up the MCU from the Stop mode when a given stimulus is received.	MX	MX	MX
		UART_HyperTerminal_DMA	UART transmission (Transmit/receive) in DMA mode between a board and a HyperTerminal PC application.	-	MX	-
		UART_HyperTerminal_IT	UART transmission (Transmit/receive) in Interrupt mode between a board and a HyperTerminal PC application.	-	MX	-
		UART_Printf	Re-routing of the C library printf function to the UART.	-	MX	-
		UART_ReceptionToIdle_CircularDMA	How to use the HAL UART API for the reception to the IDLE event in circular DMA mode.	-	-	MX
		UART_Trace_TrustZone	How to use UART to define a secure trace communication path when TrustZone security is activated (Option bit TZEN=1).	MX TrustZone	-	-
		UART_TwoBoards_ComDMA	UART transmission (Transmit/receive) in DMA mode between two boards.	-	-	MX
		UART_TwoBoards_ComIT	UART transmission (Transmit/receive) in Interrupt mode between two boards.	-	-	MX
		UART_TwoBoards_ComPolling	UART transmission (Transmit/receive) in Polling mode between two boards.	-	-	MX
		UART_WakeUpFromStopUsingFIFO	Configuration of a UART to wake up the MCU from the Stop mode with a FIFO level when a given stimulus is received.	MX	-	MX
	USART	USART_SlaveMode	This example describes USART-SPI communication (Transmit/receive) between two boards where the USART is configured as a slave.	MX	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[(1)]	STM32L552E-EV[(1)]	NUCLEO-L552ZE-Q[(1)]
Examples	WWDG	WWDG_Example	Configuration of the HAL API to periodically update the WWDG counter and simulate a software fault that generates an MCU WWDG reset when a predefined time period has elapsed.	MX	MX	MX
	Total number of examples: 213			**65**	**51**	**97**
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.	-	-	MX
		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.	-	-	MX
		ADC_ContinuousConversion_TriggerSW_LowPower_Init	How to use an ADC peripheral with ADC low-power features.	-	-	MX
		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).	-	-	MX
		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, refer to other examples).	-	-	MX
		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, refer to other examples).	-	-	MX
		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 is indefinitely transferred by DMA into a table (Circular mode).	-	-	MX
	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 (VREFINT), in interrupt mode. This example is based on the STM32L5xx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
		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 (VREFINT), in interrupt mode. This example is based on the STM32L5xx COMP LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	-	MX
		COMP_CompareGpioVsVrefInt_OutputGpio_Init	How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the internal voltage reference (VREFINT). The comparator output is connected to a GPIO. This example is based on the STM32L5xx COMP LL API.	-	-	MX
		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 (VREFINT) and a fraction of the internal voltage reference (VREFINT/2), in interrupt mode. This example is based on the STM32L5xx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK(1)	STM32L552E-EV(1)	NUCLEO-L552ZE-Q(1)
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).	-	-	MX
		CRC_UserDefinedPolynomial	How to configure and use the CRC calculation unit to compute an 8-bit CRC code for a given data buffer, based on a user-defined generating polynomial. The peripheral initialization is done using LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
	DAC	DAC_GenerateConstantSignal_TriggerSW_Init	How to use the DAC peripheral to generate a constant voltage signal. This example is based on the STM32L5xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
		DAC_GenerateConstantSignal_TriggerSW_LP_Init	How to use the DAC peripheral to generate a constant voltage signal with the DAC low-power feature sample-and-hold. 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 STM32L5xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
	DMA	DMA_CopyFromFlashToMemory	How to use a DMA channel to transfer a word data buffer from Flash memory to embedded SRAM. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	X
		DMA_CopyFromFlashToMemory_Init	How to use a DMA channel to transfer a word data buffer from Flash memory to embedded SRAM. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	-	-	MX
	EXTI	EXTI_ToggleLedOnIT_Init	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 STM32L5xx LL API. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	-	-	MX
	GPIO	GPIO_InfiniteLedToggling	How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32L5xx 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 STM32L5xx LL API. The peripheral is initialized with the LL initialization function to demonstrate LL init usage.	-	-	MX
	I^2C	I2C_OneBoard_Communication_PollingAndIT_Init	How to transmit data bytes from an I^2C master device using polling mode to an I^2C slave device using interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	-	MX
		I2C_TwoBoards_WakeUpFromStop_IT_Init	How to handle the reception of a data byte from an I^2C slave device in Stop 1 mode by an I^2C master device, both using interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	-	MX
	IWDG	IWDG_RefreshUntilUserEvent	How to configure the IWDG peripheral to ensure periodical counter update and generate an MCU IWDG reset when a User push-button is pressed. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	-	X

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples_LL	IWDG	IWDG_RefreshUntilUserEvent_Init	How to configure the IWDG peripheral to ensure periodical counter update and generate an MCU IWDG reset when a User push-button is pressed. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	-	MX
	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 STM32L5xx 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 STM32L5xx LPTIM LL API. The peripheral is initialized with the LL initialization function to demonstrate LL init usage.	-	-	MX
	LPUART	LPUART_WakeUpFromStop2_Init	Configuration of 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 the LL initialization function to demonstrate LL init usage.	-	-	MX
	PKA	PKA_ECDSA_Sign	How to use the low-layer PKA API to generate an ECDSA signature.	MX	-	-
		PKA_ModularExponentiation	How to use the low-layer PKA API to execute RSA modular exponentiation.	MX	-	-
	PWR	PWR_EnterStandbyMode	How to enter the Standby mode and wake up from this mode by using an external reset or a wake-up interrupt.	-	-	MX
		PWR_EnterStopMode	How to enter the Stop 1 mode.	-	-	MX
	RCC	RCC_OutputSystemClockOnMCO	Configuration of MCO pin (PA8) to output the system clock.	-	-	MX
		RCC_UseHSI_PLLasSystemClock	Modification of the PLL parameters in run time.	-	-	MX
	RTC	RTC_Alarm	Configuration of 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_ExitStandbyWithWakeUpTimer_Init	How to periodically enter and wake up from STANDBY mode thanks to the RTC Wake-Up Timer (WUT).	-	-	MX
		RTC_TimeStamp_Init	Configuration of the Timestamp using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
	SPI	SPI_TwoBoards_FullDuplex_IT_Master_Init	Data buffer transmission and reception via SPI using Interrupt mode. This example is based on the STM32L5xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples_LL	SPI	SPI_TwoBoards_FullDuplex_IT_Slave_Init	Data buffer transmission and reception via SPI using Interrupt mode. This example is based on the STM32L5xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
	TIM	TIM_BreakAndDeadtime	Configuration of the TIM peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined deadtime value, use the break feature, and lock the break and dead-time configuration.	-	-	X
		TIM_BreakAndDeadtime_Init	Configuration of the TIM peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined deadtime value, use the break feature, and lock the break and dead-time configuration.	-	-	MX
		TIM_DMA	Use of 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 STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	X
		TIM_DMA_Init	Use of 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 STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
		TIM_InputCapture	Use of the TIM peripheral to measure a periodic signal frequency provided either by an external signal generator or by another timer instance. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	X
		TIM_InputCapture_Init	Use of the TIM peripheral to measure a periodic signal frequency provided either by an external signal generator or by another timer instance. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
		TIM_OnePulse_Init	Configuration of a timer to generate a positive pulse in Output Compare mode with a length of tPULSE and after a delay of tDELAY. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses the LL initialization function to demonstrate LL Init.	-	-	MX
		TIM_OutputCompare	Configuration of the TIM peripheral to generate an output waveform in different output compare modes. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	X
		TIM_OutputCompare_Init	Configuration of the TIM peripheral to generate an output waveform in different output compare modes. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
		TIM_PWMOutput	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	X
		TIM_PWMOutput_Init	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses the LL initialization function to demonstrate LL Init.	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples_LL	TIM	TIM_TimeBase	Configuration of the TIM peripheral to generate a time base. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	X
		TIM_TimeBase_Init	Configuration of the TIM peripheral to generate a time base. This example is based on the STM32L5xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
	USART	USART_Communication_Rx_IT	Configuration of GPIO and USART peripherals to receive characters from a 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	This example shows how to configure GPIO and USART peripheral for continuously receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL unitary services functions for optimization purposes (Performance and size).	-	-	MX
		USART_Communication_Rx_IT_Continuous_VCP_Init	This example shows how to configure GPIO and LPUART peripheral for continuously receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL unitary services functions for optimization purposes (Performance and size).	-	-	MX
		USART_Communication_Rx_IT_Init	Configuration of GPIO and USART peripherals to receive characters from a HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization uses the LL initialization function to demonstrate LL init.	-	-	X
		USART_Communication_TxRx_DMA	Configuration of GPIO and USART peripherals to send characters asynchronously to/from a HyperTerminal (PC) in DMA mode.	-	-	X
		USART_Communication_TxRx_DMA_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to/from a HyperTerminal (PC) in DMA mode. This example is based on STM32L5xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (Performance and size).	-	-	MX
		USART_Communication_Tx_IT_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on STM32L5xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (Performance and size).	-	-	MX
		USART_Communication_Tx_IT_VCP_Init	This example shows how to configure GPIO and LPUART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on STM32L5xx LPUART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (Performance and size).	-	-	MX
		USART_Communication_Tx_Init	This example shows how to configure GPIO and USART peripherals to send characters asynchronously to a HyperTerminal (PC) in Polling mode. If the transfer could not be completed within the allocated time, a timeout allows exiting from the sequence with a Timeout error code. This example is based on STM32L5xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (Performance and size).	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[(1)]	STM32L552E-EV[(1)]	NUCLEO-L552ZE-Q[(1)]
Examples_LL	USART	USART_Communication_Tx_VCP_Init	This example shows how to configure GPIO and LPUART peripherals to send characters asynchronously to a HyperTerminal (PC) in Polling mode. If the transfer could not be completed within the allocated time, a timeout allows exiting from the sequence with a Timeout error code. This example is based on STM32L5xx LPUART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (Performance and size).	-	-	MX
		USART_HardwareFlowControl_Init	Configuration of GPIO and peripheral to receive characters asynchronously from a HyperTerminal (PC) in Interrupt mode with the Hardware Flow Control feature enabled. This example is based on STM32L5xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
		USART_SyncCommunication_FullDuplex_DMA_Init	Configuration of 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 STM32L5xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
		USART_SyncCommunication_FullDuplex_IT_Init	Configuration of 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 STM32L5xx 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	Configuration of GPIO and USART3 peripherals to allow the characters received on USART_RX pin to wake up the MCU from low-power mode.	-	-	MX
	UTILS	UTILS_ConfigureSystemClock	Use of UTILS LL API to configure the system clock using PLL with HSI as source clock.	-	-	MX
		UTILS_ReadDeviceInfo	This example reads the UID, the Device ID, and the Revision ID, and saves them into a global information buffer.	-	-	MX
	WWDG	WWDG_RefreshUntilUserEvent_Init	Configuration of the WWDG to periodically update the counter and generate an MCU WWDG reset when a user button is pressed. The peripheral initialization uses LL unitary service functions for optimization purposes (Performance and size).	-	-	MX
	Total number of examples_ll: 69			**2**	**0**	**67**
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, refer to other examples). It is based on the STM32L5xx ADC HAL and LL API. The LL API is used for performance improvement.	-	-	MX
	CRC	CRC_PolynomialUpdate	How to use the CRC peripheral through the STM32L5xx CRC HAL and LL API.	-	-	MX
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the STM32L5xx DMA HAL and LL API. The LL API is used for performance improvement.	-	-	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Examples_MIX	PWR	PWR_STOP1	How to enter the STOP 1 mode and wake up from this mode by using an external reset or a wake-up interrupt (All the RCC function calls use RCC LL API for minimizing footprint and maximizing performance).	-	-	MX
	SPI	SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	-	MX
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	-	MX
		SPI_HalfDuplex_ComPollingIT_Master	Data buffer transmission/reception between two boards via SPI using Polling (LL driver) and Interrupt modes (HAL driver).	-	-	MX
		SPI_HalfDuplex_ComPollingIT_Slave	Data buffer transmission/reception between two boards via SPI using Polling (LL driver) and Interrupt modes (HAL driver).	-	-	MX
	TIM	TIM_PWMInput	Use of the TIM peripheral to measure an external signal frequency and duty cycle.	-	-	MX
	UART	UART_HyperTerminal_IT	Use of a UART to transmit data (Transmit/receive) between a board and a HyperTerminal PC application in Interrupt mode. This example describes how to use the USART peripheral through the STM32L5xx UART HAL and LL API, which is used for performance improvement.	-	-	MX
		UART_HyperTerminal_TxPolling_RxIT	Use of a UART to transmit data (Transmit/receive) between a board and a HyperTerminal PC application both in Polling and Interrupt modes. This example describes how to use the USART peripheral through the STM32L5xx UART HAL and LL API, the LL API used for performance improvement.	-	-	MX
	Total number of examples_mix: 11			**0**	**0**	**11**
Applications	-	SBSFU	The SBSFU provides a Root of Trust solution including Secure Boot and Secure Firmware Update functionalities that are used before executing the application and provides an example of a secure service (GPIO toggle) that is isolated from the non-secure application but can be used by the non-secure application at run-time.	-	-	X TrustZone
		TFM	The TFM provides a Root of Trust solution including Secure Boot and Secure Firmware Update functionalities that are used before executing the application and provides TFM secure services that are isolated from the non-secure application but can be used by the non-secure application at run-time.	X TrustZone	-	-
	BLE	HeartRate	This application shows how to use the BLE component for the HeartRate profile application.	X	-	-
	FatFs	FatFs_MultiDrives	How to use STM32Cube firmware with FatFs middleware component as a generic FAT file system module. This example develops an application that exploits FatFs features, with multidrive (SRAM, microSD™) configurations.	-	X	-
		FatFs_RAMDisk	This application describes how to use STM32Cube firmware with FatFs middleware component as a generic FAT file system module, to develop an application exploiting FatFs offered features with a RAM disk (SRAM) drive configuration.	-	X	X

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Applications	FatFs	FatFs_RAMDisk_RTOS	This application describes how to use STM32Cube firmware with FatFs middleware component as a generic FAT file system module. This example develops an application exploiting FatFs features with a RAM disk (SRAM) drive-in RTOS mode configuration.	-	X	-
		FatFs_uSD_DMA	How to use STM32Cube 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.	-	X	-
		FatFs_uSD_RTOS	This application describes how to use STM32Cube firmware with FatFs middleware component as a generic FAT file system module, to develop an application exploiting FatFs offered features with the microSD™ disk drive configuration.	-	X	-
		FatFs_uSD_Standalone	How to use STM32Cube 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.	X	X	-
		FatFs_uSD_TrustZone	How to use FatFs stack with enabled TrustZone feature (TZEN=1).	X TrustZone	X TrustZone	-
	FreeRTOS	FreeRTOS_MPU	How to use the MPU feature of FreeRTOS.	-	X	X
		FreeRTOS_Mutexes	How to use mutexes with CMSIS RTOS API.	MX	MX	MX
		FreeRTOS_Queues	How to use message queues with CMSIS RTOS API.	-	-	MX
		FreeRTOS_SecureIOToggle_TrustZone	How to use FreeRTOS when the TrustZone feature is enabled (TZEN=1).	MX TrustZone	MX TrustZone	MX TrustZone
		FreeRTOS_Semaphore	How to use semaphores with CMSIS RTOS API.	-	-	MX
		FreeRTOS_SemaphoreFromISR	How to use semaphore from ISR with CMSIS RTOS API.	-	-	MX
		FreeRTOS_ThreadCreation	How to implement thread creation using CMSIS RTOS API.	MX	MX	MX
		FreeRTOS_Timers	How to use timers of CMSIS RTOS API.	MX	MX	MX

Level	Module Name	Project Name	Description	STM32L562E-DK[1]	STM32L552E-EV[1]	NUCLEO-L552ZE-Q[1]
Applications	TouchSensing	TouchSensing_1touchkey	Use of the STMTouch driver with 1 touch key sensor.	-	X	-
	USB-PD	USB-PD_Consumer_1port	How to create a simple type C Consumer.	MX	MX	MX
	USB_Device	AUDIO_Standalone	This application shows how to use the implementation of the audio streaming (Out: Speaker/Headset) capability on the STM32L5xx devices.	-	MX	-
		CDC_Standalone	This application describes how to use USB device application based on the Device Communication Class (CDC) following the PSTN sub-protocol on the STM32L5xx devices.	MX	MX	MX
		CustomHID_Standalone	This application shows how to use the USB device application based on the Custom HID Class on the STM32L5xx devices.	-	MX	-
		DFU_Standalone	Compliant implementation of the Device Firmware Upgrade (DFU) capability to program the embedded Flash memory through the USB peripheral.	MX	MX	MX
		HID_Standalone	Use of the USB device application based on the Human Interface (HID).	MX	MX	MX
		MSC_Standalone	This application shows how to use the USB device application based on the Mass Storage Class (MSC) on the STM32L5xx devices.	MX	MX	-
	mbedTLS	Crypto_Selftest	This application implements on STM32L562E-DK board a set of cryptographic features through mbed TLS self-test functions of individual mbed TLS components selectively chosen in a single configuration file "mbedtls_config.h".	MX	-	-
	Total number of applications: 48			**14**	**20**	**14**
Demonstrations	-	Demo	The STM32Cube demonstration platform comes on top of the STM32Cube as a firmware package that offers a full set of software components based on a modular architecture. The STM32Cube demonstration platform is built around basic UI or TouchGFX graphical interface. It is based on the STM32Cube HAL, BSP, and several middleware components.	X	X	-
	Total number of demonstrations: 2			**1**	**1**	**0**
Total number of projects: 352				**85**	**75**	**192**

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

Revision history

Table 2. Document revision history

Date	Version	Changes
14-Jan-2020	1	Initial release
11-Feb-2020	2	Updated Table 1. STM32CubeL5 firmware examples in line with STM32CubeL5 firmware V1.2.0
10-Mar-2021	3	Updated Table 1. STM32CubeL5 firmware examples in line with STM32CubeL5 firmware V1.4.0

Contents

List of tables