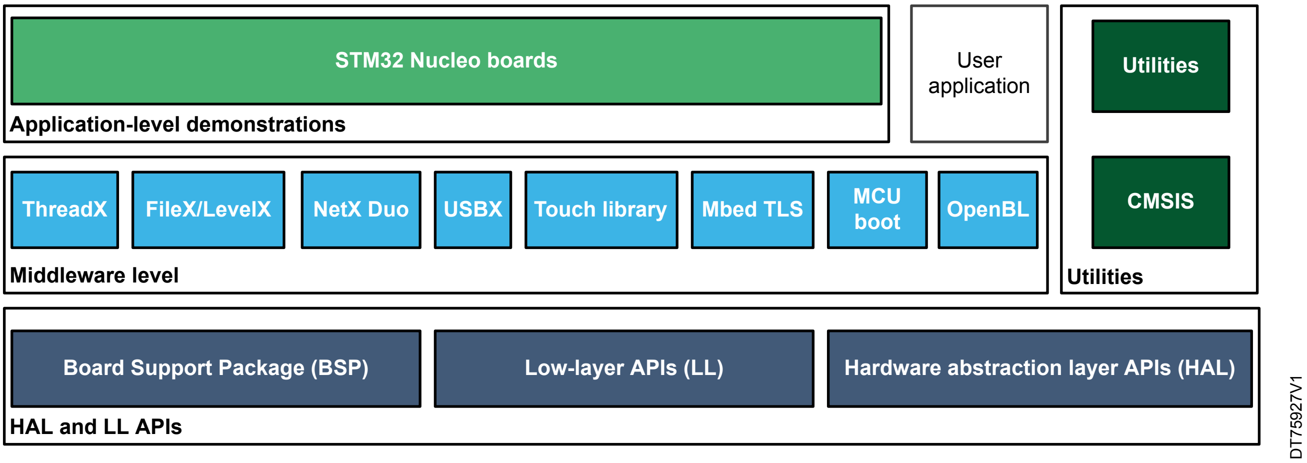
STM32Cube MCU package examples for STM32U3 series

Introduction

The STM32U3 MCU package comes with a rich set of examples running on STMicroelectronics boards. The examples are organized by board, and are provided with preconfigured projects for the main supported toolchains (see figure below).

Figure 1. **STM32CubeU3 firmware components**



1 STM32CubeU3 examples

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

- **Examples**
  The examples use only the HAL and BSP drivers (middleware components are not used). Their objective is to demonstrate the product/peripherals features and usage. They are organized per peripheral (one folder per peripheral, for example, TIM). Their complexity level ranges from the basic usage of a given peripheral (for example, PWM generation using timer) to the integration of several peripherals (for example, how to use DAC for signal generation with synchronization from TIM6 and DMA). The usage of the board resources is reduced to the strict minimum.
- **Examples_LL**
  These examples only use the LL drivers (HAL drivers and middleware components are not used). They offer an optimum implementation of typical use cases of the peripheral features and configuration sequences. The LL examples are organized per peripheral (one folder for each peripheral, for example, TIM) and run exclusively on the Nucleo board.
- **Examples_MIX**
  These examples only use HAL, BSP, and LL drivers (middleware components are not used). They aim at demonstrating how to use both HAL and LL APIs in the same application to combine the advantages of both APIs:
  - HAL drivers offer high-level function-oriented APIs, which have a high level of portability since they hide product/IP complexity to end-users.
  - LL drivers offer low-level APIs at register level with better optimization.

  The examples are organized per peripheral (one folder for each peripheral, for example, TIM) and run exclusively on the Nucleo board.
- **Applications**
  The applications demonstrate the product performance and how to use the available middleware stacks. They are organized either by middleware (one folder per middleware, for example USB host) or by product feature that requires high-level firmware bricks (for example, audio). The integration of applications that use several middleware stacks is also supported.

The examples are located under *STM32Cube_FW_U3_VX.Y.Z\Projects\*. They all have the same structure:

- *\Inc* folder, containing all header files.
- \Src folder, containing the source code.
- *\EWARM*, *\MDK-ARM*, and *\STM32CubeIDE* folders, containing the preconfigured project for each toolchain.
- readme.html 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 the preferred toolchain.
2. Rebuild all files and load the image into target memory.
3. Run the example by following the `readme.html` instructions.

Note: *Refer to sections "Development Toolchains and Compilers" and "Supported Devices and EVAL, Nucleo, and Discovery boards" of the firmware package release notes to know about the software/hardware environment used for the firmware development and validation. The correct operation of the provided examples is not guaranteed on some 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 the board, provided it has the same hardware functions (such as LED, LCD display, 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 contains the list of examples provided with the STM32U3 MCU Package.

Note: *STM32CubeMX-generated examples are highlighted with the* MX *STM32CubeMX icon. TrustZone indicates that the example is Arm® TrustZone® enabled.*
Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

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

Table 1. **STM32CubeU3 firmware examples**

STM32CubeMX-generated examples are highlighted with the icon MX.

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
ROT_ Provisioning	-	OEMiROT	This section provides an overview of the available scripts for OEMiROT boot path.	-	X
		OEMiRoT_OEMuRoT	This section provides an overview of the available scripts for OEMiROT_OEMuROT boot path.	-	X
	Total number of rot_provisioning			**0**	**2**
Templates_ Board	-	Starter project	This project provides a reference template for the NUCLEO-U385RG-Q board based on the STM32Cube HAL API and the BSP drivers that can be used to build any firmware application.	-	MX
	Total number of templates_board			**0**	**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 security is not enabled (TZEN = 0).	-	MX
		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 (TZEN = 1).	-	MX
		TrustZoneEnabled_NoIsolation	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 (TZEN = 1) and all MCU resources (such as memories and peripherals) are configured as secure.	-	X
	ROT	OEMiROT_Appli_TrustZone	This project provides a OEMiROT boot path reference template. The boot is performed through the OEMiROT boot path after checking the authenticity and integrity of the project firmware and project data images.	-	X
	Total number of templates			**0**	**4**
Templates_LL	-	TrustZoneDisabled	This project provides a reference template through the LL API that can be used to build any firmware application.	-	MX
	Total number of templates_ll			0	1
Examples	ADC	ADC_AnalogWatchdog	How to use an ADC peripheral with an ADC analog watchdog to monitor a channel and detect when the corresponding converted data is outside the window thresholds.	-	MX
		ADC_ContinuousConversion_TriggerSW_LowPower	How to use an ADC to convert a single channel using the ADC low-power Auto wait feature.	-	MX
		ADC_DiscontinuousConversion_TriggerSW	How to use an ADC to perform multiple conversions from different ADC channels, one at a time, after each software trigger.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples	ADC	ADC_FixedTriggerLatency	How to use an ADC to perform a single ADC conversion on a channel without any uncertainty (fixed trigger latency), at each trigger event from a timer.	-	MX
		ADC_SingleConversion_TriggerSW_IT	How to use ADC to convert a single channel at each software start. The conversion is performed using the interrupt programming model.	-	MX
		ADC_SingleConversion_TriggerTimer_DMA	How to use an ADC to perform a single ADC conversion on a channel at each trigger event from a timer. The converted data is transferred by DMA into a table in RAM.	-	MX
		ADC_TemperatureSensor	How to use an ADC to perform a single ADC conversion on the internal temperature sensor, and calculate the temperature in degrees Celsius.	-	MX
	ADF	ADF_AudioRecorder	How to configure the ADF to perform PCM recording in DMA circular mode.	-	MX
		ADF_AudioRecorder_LowPower	How to configure the ADF to perform PCM recording in low-power Stop 1 mode.	-	MX
		ADF_AudioSoundDetector	How to configure the ADF to perform audio sound detection.	-	MX
		ADF_AudioSoundDetector_LowPower	How to configure the ADF to perform audio sound detection in low-power Stop 1 mode.	-	MX
	CCB	CCB_Protected_ECCScalarMul_BlobCreation	How to use the CCB to create a blob for ECC scalar multiplication.	-	MX
		CCB_Protected_ECCScalarMul_BlobUse	How to use the CCB to compute k x P scalar multiplication, using a special blob called *ECC key blob*.	-	MX
		CCB_Protected_ECDSA_BlobCreation	How to use the CCB to create a blob for the Elliptic curve digital signature algorithm (ECDSA).	-	MX
		CCB_Protected_ECDSA_PublicKeyComputation	How to use the CCB to compute a public key, using a special blob called *ECDSA key blob*.	-	MX
		CCB_Protected_ECDSA_Signature	How to use the CCB to compute a signed message regarding the Elliptic curve digital signature algorithm, using a special blob called *ECDSA key blob*.	-	X
		CCB_Protected_RSAModularExp_BlobCreation	How to use the CCB to create a blob for RSA modular exponentiation.	-	MX
		CCB_Protected_RSAModularExp_BlobUse	How to use the CCB to create and use an RSA modular exponentiation blob.	-	MX
	COMP	COMP_CompareGpioVsVrefInt_IT	How to use a comparator to compare a voltage level applied on a GPIO pin to the internal voltage reference (V_{REFINT}), in interrupt mode.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples	COMP	COMP_CompareGpioVsVrefInt_IT_SystemStopMode	How to use a comparator to compare a voltage level applied on a GPIO pin to the internal voltage reference (V_{REFINT}), in interrupt mode and during Stop mode.	-	MX
		COMP_CompareGpioVsVrefInt_OutputGpio	How to use a comparator to compare a voltage level applied on a GPIO pin to the internal voltage reference (V_{REFINT}).	-	MX
		COMP_CompareGpioVsVrefInt_Window_IT	How to use a pair of comparators 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	Presentation of the MPU features: this example configures the MPU attributes of different MPU regions, then configures a memory area as privileged read only, and attempts to perform read and write operations in different modes.	-	MX
		CORTEXM_ModePrivilege	How to modify the Thread mode privilege access and stack. The Thread mode is entered on reset or when returning from an exception.	-	MX
		CORTEXM_ProcessStack	How to modify the Thread mode stack. The Thread mode is entered on reset, or when returning from an exception.	-	MX
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LED.	-	MX
	CRC	CRC_Bytes_Stream_7bit_CRC	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes 7-bit CRC codes derived from buffers of 8-bit data (bytes). The user-defined generating polynomial is manually set to 0x65, that is, $X^7 + X^6 + X^5 + X^2 + 1$, as used in the Train Communication Network *IEC 60870-5[17]*.	-	MX
		CRC_Data_Reversing_16bit_CRC	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes a 16-bit CRC code derived from a buffer of 32-bit data (words). The input and output data reversal features are enabled. The user-defined generating polynomial is manually set to 0x1021, that is, $X^{16} + X^{12} + X^5 + 1$ which is the CRC-CCITT generating polynomial.	-	MX
		CRC_Example	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the CRC code of a given buffer of 32-bit data words, using a fixed generator polynomial (0x4C11DB7).	-	MX
		CRC_UserDefinedPolynomial	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the 8-bit CRC code for a given buffer of 32-bit data words, based on a user-defined generating polynomial.	-	MX
	CRYP	CRYP_AES_GCM	How to use the CRYP to encrypt and decrypt data using the AES with Galois/Counter mode (GCM).	-	MX
		CRYP_DMA	How to use the AES 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 to encrypt data and generate authentication tags in GCM/GMAC/CCM modes.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples	CRYP	CRYP_GCM_GMAC_CMAC_Suspension	How to use the CRYP AES to suspend then resume the AES GCM and GMAC CMAC processing of a message, to carry out the encryption, decryption, or authentication tag computation of a higher-priority message (CMAC).	-	MX
		CRYP_SAES_ECB_CBC	How to use the secure AES coprocessor (SAES) to encrypt and decrypt data using AES ECB and CBC algorithms.	-	MX
		CRYP_SAES_SharedKey	How to use the secure AES coprocessor (SAES) to share application keys with the AES.	-	MX
		CRYP_SAES_WrapKey	How to use the secure AES coprocessor (SAES) to wrap application keys using a hardware secret key (DHUK), then use it to encrypt in polling mode.	-	MX
	DAC	DAC_ContinuousConversionDMA_TriggerEXTI	How to use the DAC to generate a signal using the DMA and triggered by an EXTI signal.	-	MX
		DAC_GenerateConstantSignal_SystemStopMode	How to use the DAC to perform a simple conversion with the MCU in Stop mode.	-	MX
		DAC_SignalsGeneration	How to use the DAC to generate several signals using the DMA controller and the DAC internal wave generator.	-	MX
		DAC_SimpleConversion	How to use the DAC to perform a simple conversion.	-	MX
	DMA	DMA_DataHandling	How to use the DMA controller to perform data handling between transferred data from the source, and transfer it to the destination through the HAL API.	-	MX
		DMA_FLASHToRAM	How to use the DMA to transfer a word data buffer from flash memory to embedded SRAM, through the HAL API.	-	MX
		DMA_LinkedList	How to use the DMA to perform a list of transfers. The transfer list is organized as linked-list: each time the current transfer ends, the DMA automatically reloads the next transfer parameters, and starts it (without CPU intervention).	-	MX
		DMA_RepeatedBlock	How to configure and use the DMA HAL API to perform repeated block transactions.	-	MX
		DMA_Trigger	How to configure and use the DMA HAL API to perform DMA triggered transactions.	-	MX
	FDCAN	FDCAN_Adaptive_Bitrate_Receiver	How to configure the FDCAN to adapt to different CAN bit rates using restricted mode.	-	MX
		FDCAN_Adaptive_Bitrate_Transmitter	How to configure the FDCAN to adapt to different CAN bit rates using restricted mode.	-	MX
		FDCAN_Classic_Frame_Networking	How to configure the FDCAN to send and receive classic CAN frames between two FDCAN units.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples	FDCAN	FDCAN_Com_IT	How to configure the FDCAN to achieve interrupt process communication between two FDCAN units.	-	MX
		FDCAN_Com_Polling	How to configure the FDCAN to achieve a polling process communication between two FDCAN units.	-	MX
		FDCAN_Loopback	How to configure the FDCAN to operate in loopback mode.	-	MX
		FDCAN_Power_Down	This example describes the FDCAN operation in power-down mode.	-	MX
	FLASH	FLASH_ChangeOptionBytes	How to configure and use the FLASH HAL API to change option byte values.	-	MX
		FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal flash memory.	-	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 (TZEN = 1).	-	MX
		FLASH_HideProtection_TrustZone	How to configure and use the FLASH HAL API for the secure hide protection and extended secure hide protection of the internal flash memory.	-	X
		FLASH_SwapBanks	Guide through the configuration steps to program the internal flash memory bank 1 and bank 2, and swap between both banks by means of the FLASH HAL API.	-	X
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	-	MX
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	-	MX
		GPIO_IOToggle_TrustZone	How to use the HAL GPIO to toggle secure and nonsecure I/Os when TrustZone® security is activated (TZEN = 1).	-	MX
	GTZC	GTZC_TZSC_MPCBB_TrustZone	How to use the HAL GTZC MPCBB to build any example with illegal access detection when TrustZone® security is activated (TZEN option byte = 0xB4).	-	MX
	HAL	HAL_TimeBase_RTC_WKUP	How to customize the HAL using RTC wake-up as the main source of time base, instead of the SysTick.	-	MX
		HAL_TimeBase_TIM	How to customize the HAL using a general-purpose timer as the main source of time base instead of the SysTick.	-	MX
	HASH	HASH_HMAC_SHA224SHA1_DMA_Suspension	How to suspend the HMAC digest computation when data are fed to the HASH by DMA.	-	MX
		HASH_SHA256	How to use the HASH to hash data using the SHA256 algorithm.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW(1)	NUCLEO-U385RG-Q
Examples	HASH	HASH_SHA384	How to use the HASH to hash data using the SHA384 algorithm.	-	MX
		HASH_SHA512	How to use the HASH to hash data using SHA512 algorithms.	-	MX
	I2C	I2C_Sensor_Private_Command_IT	How to handle I^{2}C data buffer transmission/reception between the NUCLEO-U385RG-Q and the X-NUCLEO-IKS4A1 boards, using an interrupt.	-	MX
		I2C_TwoBoards_AdvComIT	How to handle several I^{2}C data buffer transmission/reception between a master and a slave, using an interrupt.	-	MX
		I2C_TwoBoards_ComDMA	How to handle an I2C to perform a data buffer transmission/reception between two boards, using the DMA.	-	MX
		I2C_TwoBoards_ComDMA_Autonomous_Master	How to handle an I2C as a master to perform a data buffer transmission/reception in autonomous mode between two boards, using the DMA.	-	MX
		I2C_TwoBoards_ComDMA_Autonomous_Slave	How to handle an I2C as a slave to perform a data buffer transmission/reception in autonomous mode between two boards, using the DMA.	-	MX
		I2C_TwoBoards_ComIT	How to handle I^{2}C data buffer transmission/reception between two boards, using an interrupt.	-	MX
		I2C_TwoBoards_MultiMasterIT_Master	How to handle an I2C as a master to perform a data buffer communication between two boards, using an interrupt, two masters, and one slave.	-	MX
		I2C_TwoBoards_MultiMasterIT_Slave	How to handle an I2C as a slave to perform a data buffer communication between two boards, using an interrupt, two masters, and one slave.	-	MX
		I2C_TwoBoards_RestartAdvComIT	How to perform multiple I^{2}C data buffer transmission/reception between two boards, in interrupt mode, and with a restart condition.	-	MX
		I2C_TwoBoards_RestartComIT	How to handle single I^{2}C data buffer transmission/reception between two boards, in interrupt mode, and with a restart condition.	-	MX
		I2C_WakeUpFromStop	How to handle I^{2}C data buffer transmission/reception between two boards, using an interrupt when the device is in Stop mode.	-	MX
	I3C	I3C_Controller_Direct_Command_DMA	How to handle a direct command procedure between an I3C controller and an I3C target, using the DMA.	-	MX
		I3C_Controller_ENTDAA_IT	How to handle an ENTDAA procedure between an I3C controller and one or more I3C targets.	-	MX
		I3C_Controller_HotJoin_IT	How to handle an I3C as a target to perform a HOTJOIN procedure between an I3C controller and I3C targets, using interrupt.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW(1)	NUCLEO-U385RG-Q
Examples	I3C	I3C_Controller_I2C_ComDMA	How to handle I^{2}C communication as I3C controller data buffer transmission/ reception between two boards, using the DMA.	-	MX
		I3C_Controller_IBI_Wakeup_IT	How to handle an I3C as a controller to generate an in-band interrupt event, and exchange it between an I3C controller in low-power mode and I3C targets.	-	MX
		I3C_Controller_InBandInterrupt_IT	How to handle an I3C as a controller to generate an in-band interrupt event, and exchange it between an I3C controller and I3C targets.	-	MX
		I3C_Controller_Private_Autonomous	How to handle an I3C controller to transmit a private message in autonomous mode, using the NUCLEO-U385RG-Q SCL pin (PB13, Morpho CN10 pin 30).	-	MX
		I3C_Controller_Private_Command_IT	How to handle an I3C as a controller to perform data buffer transmission/ reception between two boards, using an interrupt.	-	MX
		I3C_Controller_ResetPattern_RSTACT	How to handle an I3C as a controller to configure a selected reset pattern after a Direct RSTACT Command procedure between an I3C controller and an I3C target, using an interrupt.	-	MX
		I3C_Controller_ResetPattern_WakeUpFromLowPower	How to handle an I3C as a controller to send a reset pattern, when the target is in low-power mode.	-	MX
		I3C_Controller_Switch_To_Target	How to handle an I3C as a controller to perform a Controller Role Request Direct Command procedure between an I3C controller and an I3C target, using an interrupt.	-	MX
		I3C_Controller_WakeUpFromStop	How to handle an I3C as a controller to perform data buffer transmission/ reception between two boards, using an interrupt, when the target is in Stop mode.	-	MX
		I3C_Sensor_Direct_Command_DMA	How to handle a Direct Command procedure between the NUCLEO-U385RG-Q and the X-NUCLEO-IKS4A1 boards, using the DMA.	-	MX
		I3C_Sensor_Private_Command_IT	How to handle an I3C as a controller to perform data buffer transmission/ reception between the NUCLEO-U385RG-Q and the X-NUCLEO-IKS4A1 boards, using an interrupt.	-	MX
		I3C_Target_Direct_Command_DMA	How to handle a Direct Command procedure between an I3C controller and an I3C target, using the controller in DMA mode.	-	MX
		I3C_Target_ENTDAA_IT	How to handle an ENTDAA procedure between an I3C controller and one or more I3C targets.	-	MX
		I3C_Target_HotJoin_IT	How to handle an I3C as a controller to perform a HOTJOIN procedure between an I3C controller and I3C targets, using an interrupt.	-	MX
		I3C_Target_I2C_ComDMA	How to handle I^{2}C data buffer transmission/reception between two boards, using the DMA.	-	MX
		I3C_Target_IBI_Wakeup_IT	How to handle an I3C as a target to generate an in-band interrupt event and exchange it with an I3C controller in Stop mode.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[(1)]	NUCLEO-U385RG-Q
Examples	I3C	I3C_Target_InBandInterrupt_IT	How to handle an I3C as a target to generate an in-band interrupt event and exchange it with an I3C controller.	-	MX
		I3C_Target_Private_Autonomous	How to handle an I3C as a target to perform data buffer transmission/reception between two boards, in autonomous mode.	-	MX
		I3C_Target_Private_Command_IT	How to handle an I3C as a target to perform data buffer transmission/reception between two boards, using an interrupt.	-	MX
		I3C_Target_ResetPattern_RSTACT	How to handle an I3C as a target to configure a selected reset pattern after a Direct RSTACT Command procedure between an I3C controller and an I3C target, using an interrupt.	-	MX
		I3C_Target_ResetPattern_WakeUpFromStandBy	How to handle an I3C as a target to wake up from Standby when the target receives a reset pattern.	-	MX
		I3C_Target_ResetPattern_WakeUpFromStop	How to handle an I3C as a target to wake up from Stop when the target receives a reset pattern.	-	MX
		I3C_Target_Switch_To_Controller	How to handle an I3C as a target to perform a Controller Role Request procedure to an I3C controller.	-	MX
		I3C_Target_WakeUpFromStop	How to handle an I3C as a target to perform data buffer transmission/reception between two boards, using an interrupt, when the target is in Stop mode.	-	MX
	ICACHE	ICACHE_Memory_Remap	How to execute code from a remapped region configured through the ICACHE HAL driver.	MX	-
	IWDG	IWDG_WindowMode	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_PWM_LSE	How to configure and use, through the HAL LPTIM API, the LPTIM to generate a PWM signal in a low-power mode. The LSE is used as a counter clock.	-	MX
		LPTIM_PulseCounter	How to configure and use, through the LPTIM HAL API, the LPTIM to count pulses.	-	MX
		LPTIM_Timeout	How to implement, through the HAL LPTIM API, a timeout with the LPTIM to wake up the system from a low-power mode.	-	MX
	OPAMP	OPAMP_Follower	How to configure the OPAMP in follower mode, interconnected with DAC and COMP.	-	MX
		OPAMP_PGA	How to use the built-in PGA mode (OPAMP programmable gain).	-	MX
	PKA	PKA_ECDSA_Sign	How to compute a signed message regarding the Elliptic curve digital signature algorithm (ECDSA).	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[(1)]	NUCLEO-U385RG-Q
Examples	PKA	PKA_ECDSA_Verify	How to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA).	-	MX
		PKA_ModularExponentiation	How to use the PKA to execute modular exponentiation. This allows ciphering/ deciphering a text.	-	MX
	PWR	PWR_ModesSelection	How to configure the system to measure the current consumption in different low-power modes.	-	MX
		PWR_RUN_SMPS	How to use the SMPS PWR regulator.	-	MX
		PWR_SHUTDOWN	How to enter the system in Shutdown mode and wake up from this mode using an external reset or the WKUP pin.	-	MX
		PWR_SHUTDOWN_RTC	How to enter the system in Shutdown mode and wake up from this mode using the RTC internal wake-up interrupt.	-	MX
		PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	-	MX
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using an external reset or the WKUP pin.	-	MX
		PWR_STANDBY_RTC	How to enter the Standby mode and wake up from this mode by using an external reset or the RTC wake-up timer.	-	MX
		PWR_STOP0	How to enter Stop 0 mode and wake up from this mode using an interrupt.	-	MX
		PWR_STOP0_RTC	How to enter Stop 0 mode and wake up from this mode using an interrupt from the RTC wake-up timer.	-	MX
		PWR_STOP1	How to enter Stop 1 mode and wake up from this mode using an interrupt.	-	MX
		PWR_STOP1_RTC	How to enter the Stop 1 mode and wake up from this mode by using the RTC wake-up timer.	-	MX
		PWR_STOP2	How to enter the Stop 2 mode and wake up from this mode using an external reset or a wake-up interrupt.	-	MX
		PWR_STOP2_RTC	How to enter the Stop 2 mode and wake up from this mode using an external reset or the RTC wake-up timer.	-	MX
		PWR_STOP3	How to enter the Stop 3 mode and wake up from this mode using an external reset or a wake-up interrupt.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[(1)]	NUCLEO-U385RG-Q
Examples	PWR	PWR_STOP3_RTC	How to enter the Stop 3 mode and wake up from this mode using the RTC internal wake-up interrupt.	-	MX
	RAMCFG	RAMCFG_Parity_Error	How to configure and use the RAMCFG HAL API to enable parity error detection and generate parity error interrupt.	-	MX
		RAMCFG_WriteProtection	How to configure and use the RAMCFG HAL API to configure RAMCFG SRAM write protection page.	-	MX
	RCC	RCC_CRS_Synchronization_IT	How to configure the clock recovery system (CRS) in interrupt mode, using the RCC HAL API.	-	MX
		RCC_ClockConfig	How to configure the system clock (SYSCLK) and modify the clock settings in Run mode, using the RCC HAL API.	-	MX
		RCC_LSEConfig	How to enabler/disable the low-speed external (LSE) RC oscillator (about 32 KHz) at runtime, using the RCC HAL API.	-	MX
		RCC_LSIConfig	How to enable/disable the low-speed internal (LSI) RC oscillator (about 32 KHz) at runtime, using the RCC HAL API.	-	MX
		RCC_MSIConfig	How to change the multispeed internal (MSI) RC oscillator frequency at runtime, using the PWR and RCC HAL API.	-	MX
	RNG	RNG_MultiRNG_IT	How to configure the RNG using the HAL API. This example uses RNG interrupts to generate 32-bit long random numbers.	-	MX
	RTC	RTC_Alarm	How to configure and generate an RTC alarm using the RTC HAL API.	-	MX
		RTC_LowPower_STANDBY_WUT	How to periodically enter and wake up from Standby mode thanks to the RTC wake-up timer (WUT).	-	MX
		RTC_Tamper	How to configure the tamper detection with backup registers erase.	-	MX
		RTC_TimeStamp	How to configure the RTC HAL API to demonstrate the timestamp feature.	-	MX
		RTC_TrustZone	How to configure the TrustZone®-aware RTC peripheral when TrustZone® security is activated (TZEN = 1): some features of the RTC can be secure while the others are nonsecure.	-	MX
	SAI	SAI_AudioLoopback	How to configure the SAI to perform audio loopback (record then playback).	-	MX
		SAI_AudioLoopback_LowPower	How to configure the SAI to perform audio loopback in low-power sleep mode.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples	SAI	SAI_AudioPlay	How to configure the SAI to perform audio playback in DMA circular mode.	-	MX
		SAI_AudioRecord_PDM	How to configure the SAI to perform PDM record in DMA circular mode.	-	MX
		SAI_AudioRecord_PDM_LowPower	How to configure the SAI to perform PDM record in low-power sleep mode.	-	MX
		SAI_DigitalLoopback	How to configure the SAI to perform transmission/reception in DMA circular mode.	-	MX
	SD	SD_ReadWrite_DMA	How to perform some write and read transfers to an SD card with the SDMMC operating in internal DMA mode.	-	MX
	SMBUS	SMBUS_TwoBoards_ComIT_Autonomous_Master	How to handle SMBUS data buffer transmission/reception between two boards, using the autonomous mode.	-	MX
		SMBUS_TwoBoards_ComIT_Autonomous_Slave	How to handle SMBUS data buffer transmission/reception between two boards, using the autonomous mode.	-	MX
	SPI	SPI_FullDuplex_ComDMA_Autonomous_Master	How to handle an SPI as a master to perform data buffer transmission/reception in autonomous mode between two boards, using the DMA in autonomous mode.	-	MX
		SPI_FullDuplex_ComDMA_Autonomous_Slave	How to handle an SPI as a slave to perform data buffer transmission/reception in autonomous mode between two boards, using the DMA in autonomous mode.	-	MX
		SPI_FullDuplex_ComDMA_LowPower_Master	How to handle an SPI so that a data buffer transmission/reception in DMA mode wakes up the slave board from low-power mode.	-	MX
		SPI_FullDuplex_ComDMA_LowPower_Slave	How to handle an SPI so that a data buffer transmission/reception in DMA mode wakes up the slave board from low-power mode.	-	MX
		SPI_FullDuplex_ComDMA_Master	How to handle an SPI as a master to perform a data buffer transmission/reception between two boards, using the DMA.	-	MX
		SPI_FullDuplex_ComDMA_Slave	How to handle an SPI as a slave to perform a data buffer transmission/reception between two boards, using the DMA.	-	MX
		SPI_FullDuplex_ComIT_Master	How to handle a SPI as a master to perform data buffer transmission/reception between two boards in interrupt mode.	-	MX
		SPI_FullDuplex_ComIT_Slave	How to handle a SPI as a slave to perform data buffer transmission/reception between two boards, in interrupt mode.	-	MX
		SPI_FullDuplex_ComPolling_Master	How to handle a SPI as a master to perform data buffer transmission/reception between two boards, in polling mode.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples	SPI	SPI_FullDuplex_ComPolling_Slave	How to handle a SPI as a slave to perform data buffer transmission/reception between two boards, in polling mode.	-	MX
	TIM	TIM_InputCapture_DMA	How to measure the LSI clock frequency using the DMA interface of the TIM16 timer instance.	-	MX
		TIM_OCToggle	How to configure the TIM to generate four different signals at four different frequencies.	-	MX
		TIM_PWMInput	How to use the TIM to measure the frequency and duty cycle of an external signal.	-	MX
		TIM_PWMOutput	How to configure the TIM in PWM (pulse width modulation) mode.	-	MX
		TIM_WakeUpFromSleep	How to enter the Sleep mode and wake up from this mode by using the timer update interrupt.	-	MX
	UART	LPUART_TwoBoards_ComIT	How to handle LPUART transmission (transmit/receive) in interrupt mode between two boards.	-	MX
		LPUART_WakeUpFromStop	How to configure an LPUART to wake up the MCU from Stop mode when a given stimulus is received.	-	MX
		UART_Console	How to use the HAL UART API for UART transmission (printf/getchar) via a console to interact with the user.	-	MX
		UART_HyperTerminal_DMA	How to perform UART transmission (transmit/receive) in DMA mode between a board and a HyperTerminal PC application.	-	MX
		UART_HyperTerminal_IT	How to perform UART transmission (transmit/receive) in interrupt mode between a board and a HyperTerminal PC application.	-	MX
		UART_LowPower_HyperTerminal_DMA	How to perform LPUART transmission (transmit/receive) in DMA mode between a board and a HyperTerminal PC application.	-	MX
		UART_Printf	How to reroute the C library printf function to the UART.	-	MX
		UART_ReceptionToIdle_CircularDMA	How to use the HAL UART API so that the reception is idle by an event in circular DMA mode.	-	MX
		UART_TwoBoards_ComDMA	How to perform UART transmission (transmit/receive) in DMA mode between two boards.	-	MX
		UART_TwoBoards_ComDMAlinkedlist	How to perform UART transmission (transmit/receive) in DMA mode using linkedlist between two boards.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW(1)	NUCLEO-U385RG-Q
Examples	UART	UART_TwoBoards_ComIT	How to perform UART transmission (transmit/receive) in interrupt mode between two boards.	-	MX
		UART_TwoBoards_ComPolling	How to perform UART transmission (transmit/receive) in polling mode between two boards.	-	MX
	USART	USART_SlaveMode	How to handle USART-SPI communication (transmit/receive) between two boards where the USART is configured as a slave.	-	MX
		USART_SlaveMode_DMA	How to handle USART-SPI communication (transmit/receive) with DMA between two boards where the USART is configured as a slave.	-	MX
	WWDG	WWDG_Example	How to configure the HAL API to periodically update the WWDG counter and simulate a software fault that generates an MCU WWDG reset when a predefined time period has elapsed.	-	MX
	XSPI	XSPI_EEPROM_MemoryMapped	How to erase part of the QUADSPI EEPROM, write data in interrupt mode, and access the QUADSPI EEPROM in memory-mapped mode to check the data in a forever loop.	MX	-
		XSPI_EEPROM_ReadWrite_DMA	How to erase part of a QUADSPI EEPROM, write data in DMA mode, read data in DMA mode, and compare the result in an infinite loop.	MX	-
		XSPI_NOR_ExecuteInPlace	How to execute a part of the code from the XSPI memory. To do this, a section is created where the function is stored.	MX	-
		XSPI_NOR_MemoryMapped	How to erase part of the QUADSPI memory, write data in interrupt mode, and access the QUADSPI memory in memory-mapped mode to check the data in a forever loop.	MX	-
	Total number of examples			**5**	**182**
Examples_LL	ADC	ADC_AnalogWatchdog_Init	How to use an ADC with an ADC analog watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds.	-	MX
		ADC_MultiChannelSingleConversion_Init	How to use an ADC to convert several channels. ADC conversions are performed successively in a scan sequence.	-	MX
		ADC_Oversampling_Init	How to use an ADC with oversampling.	-	MX
		ADC_SingleConversion_TriggerSW_IT_Init	How to use an ADC to convert a single channel at each software start. ADC conversions are performed using the interrupt programming model.	-	MX
		ADC_SingleConversion_TriggerTimer_DMA_Init	How to use an ADC to perform a single ADC conversion on a channel at each trigger event from a timer. The converted data is transferred by DMA into a table in RAM.	-	MX
	COMP	COMP_CompareGpioVsVrefInt_IT_Init	How to use a comparator to compare a voltage level applied on a GPIO pin to the internal voltage reference (V_{REFINT}), in interrupt mode.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples_LL	COMP	COMP_CompareGpioVsVrefInt_OutputGpio_Init	How to use a comparator to compare a voltage level applied on a GPIO pin to the internal voltage reference (V_{REFINT}) with the comparator output connected to a GPIO pin.	-	MX
		COMP_CompareGpioVsVrefInt_Window_IT_Init	How to use a pair of comparators 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	CORTEX_MPU	Presentation of the MPU features: this example configures the MPU attributes of different MPU regions, then configures a memory area as privileged read only, and attempts to perform read and write operations in different modes.	-	MX
	CRC	CRC_CalculateAndCheck	How to configure the CRC calculation unit to compute a CRC code for a given data buffer, based on a fixed generator polynomial (default value 0x4C11DB7). The peripheral initialization is done using LL unitary service functions for optimization purposes (performance and size).	-	MX
		CRC_UserDefinedPolynomial	How to configure and use the CRC calculation unit to compute an 8-bit CRC code for a given data buffer, based on a user-defined generating polynomial. The peripheral initialization is done using LL unitary service functions for optimization purposes (performance and size).	-	MX
	CRS	CRS_Synchronization_IT	How to configure the clock recovery system in interrupt mode through the STM32U3xx CRS LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		CRS_Synchronization_Polling	How to configure the clock recovery system in polling mode through the STM32U3xx CRS LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
	DMA	DMA_CopyFromFlashToMemory_Init	How to use a DMA channel to transfer a word data buffer from flash memory to an 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 STM32U3xx LL API. The peripheral initialization is done using the LL initialization function to demonstrate LL init usage.	-	MX
	GPIO	GPIO_InfiniteLedToggling_Init	How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32U3xx LL API. The peripheral is initialized with the LL initialization function to demonstrate LL init usage.	-	MX
	I2C	I2C_OneBoard_Communication_IT_Init	How to handle the reception of one data byte from an I2C slave device by an I2C master device. Both devices operate in interrupt mode. The peripheral is initialized with the LL initialization function to demonstrate LL init usage.	-	MX
		I2C_OneBoard_Communication_PollingAndIT_Init	How to transmit data bytes from an I2C master device using polling mode to an I2C slave device using interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples_LL	I2C	I2C_TwoBoards_MasterRx_SlaveTx_IT_Init	How to handle the reception of one data byte from an I2C slave device by an I2C master device. Both devices operate in interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	MX
		I2C_TwoBoards_MasterTx_SlaveRx_DMA_Init	How to transmit data bytes from an I2C master device using DMA mode to an I2C slave device using DMA mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	MX
		I2C_TwoBoards_MasterTx_SlaveRx_Init	How to transmit data bytes from an I2C master device using polling mode to an I2C slave device using interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	MX
		I2C_TwoBoards_WakeUpFromStop2_IT_Init	How to handle the reception of a data byte from an I2C slave device in Stop 2 mode by an I2C master device, both 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 I2C slave device in Stop 0 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
	I3C	I3C_Controller_Direct_Command_IT	How to handle a Direct Command procedure between an I3C controller and an I3C target, using an interrupt.	-	MX
		I3C_Controller_Direct_Command_Polling	How to handle a Direct Command procedure between an I3C controller and an I3C target, in polling mode.	-	MX
		I3C_Controller_Private_Command_IT	How to handle an I3C as a controller to handle data buffer transmission/reception between two boards, using an interrupt.	-	MX
		I3C_Target_Direct_Command_IT	How to handle a Direct Command procedure between an I3C controller and an I3C target, with the controller in interrupt mode.	-	MX
		I3C_Target_Direct_Command_Polling	How to handle a Direct Command procedure between an I3C controller and an I3C target, with the controller in polling mode.	-	MX
		I3C_Target_Private_Command_IT	How to handle an I3C as a target to handle data buffer transmission/reception between two boards, using an interrupt.	-	MX
	IWDG	IWDG_RefreshUntilUserEvent_Init	How to configure the IWDG 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
	OPAMP	OPAMP_Follower_Init	How to use the OPAMP in follower mode interconnected with DAC.	-	MX
		OPAMP_PGA_Init	How to use the OPAMP in PGA mode (OPAMP programmable gain) with a DAC.	-	MX
	PKA	PKA_ECDSA_Sign	How to use the low-layer PKA API to generate an ECDSA signature.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples_LL	RCC	RCC_OutputSystemClockOnMCO	How to configure the MCO pin (PA8) to output the system clock.	-	MX
		RCC_UseHSEasSystemClock	How to use the RCC LL API to start the HSE and use it as the system clock.	-	MX
	RNG	RNG_GenerateRandomNumbers	How to configure the RNG to generate 32-bit long random numbers. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		RNG_GenerateRandomNumbers_IT	How to configure the RNG to generate 32-bit long random numbers using interrupts. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
	RTC	RTC_Alarm_Init	How to configure the RTC LL API to configure and generate an alarm using the RTC peripheral. The peripheral initialization uses the LL initialization function.	-	MX
		RTC_Calendar_Init	How to configure the LL API to set the RTC calendar. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		RTC_ExitStandbyWithWakeUpTimer_Init	How to periodically enter and wake up from Standby mode thanks to the RTC wake-up timer (WUT).	-	MX
		RTC_Tamper_Init	How to configure the tamper using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		RTC_TimeStamp_Init	How to configure the timestamp using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
	SPI	SPI_OneBoard_FullDuplex_IT	How to perform SPI data buffer transmission/reception between two instances in the same board by using interruptions.	-	MX
		SPI_OneBoard_HalfDuplex_DMA_Init	How to configure GPIO and SPI peripherals to transmit bytes from an SPI master device to an SPI slave device in DMA mode. This example is based on the STM32U3xx SPI LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	MX
		SPI_OneBoard_HalfDuplex_IT_Init	How to configure GPIO and SPI peripherals to transmit bytes from an SPI master device to an SPI slave device in interrupt mode. This example is based on the STM32U3xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		SPI_TwoBoards_FullDuplex_DMA_Master_Init	How to perform data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32U3xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples_LL	SPI	SPI_TwoBoards_FullDuplex_DMA_Slave_Init	How to perform data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32U3xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		SPI_TwoBoards_FullDuplex_IT_Master_Init	How to perform data buffer transmission and reception via SPI using interrupt mode. This example is based on the STM32U3xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		SPI_TwoBoards_FullDuplex_IT_Slave_Init	How to perform data buffer transmission and reception via SPI using interrupt mode. This example is based on the STM32U3xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
	TIM	TIM_BreakAndDeadtime_Init	How to configure the TIM 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_InputCapture_Init	How to use the TIM to measure a periodic signal frequency provided either by an external signal generator or by another timer instance. This example is based on the STM32U3xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		TIM_OnePulse_Init	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 STM32U3xx TIM LL API. The peripheral initialization uses the LL initialization function to demonstrate LL Init.	-	MX
		TIM_OutputCompare_Init	How to configure the TIM to generate an output waveform in different output compare modes. This example is based on the STM32U3xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX
		TIM_PWMOutput_Init	How to use of a timer to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32U3xx TIM LL API. The peripheral initialization uses the LL initialization function to demonstrate LL Init.	-	MX
		TIM_TimeBase_Init	How to configure the TIM to generate a timebase. This example is based on the STM32U3xx 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	How to configure GPIO and USART peripheral to continuously receive characters from a HyperTerminal (PC) in asynchronous mode, using the interrupt mode. The peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	MX
		USART_Communication_Rx_IT_Continuous_VCP_Init	How to configure GPIO and USART peripherals to continuously receive characters from a HyperTerminal (PC) in asynchronous mode, using the interrupt mode. The peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[(1)]	NUCLEO-U385RG-Q
Examples_LL	USART	USART_Communication_Rx_IT_Init	How to configure GPIO and USART peripherals to receive characters from a HyperTerminal (PC) in asynchronous mode, using the interrupt mode. The peripheral initialization is done using the LL initialization function to demonstrate LL init usage.	-	MX
		USART_Communication_Rx_IT_VCP_Init	How to configure GPIO and USART peripherals to receive characters from a HyperTerminal (PC) in asynchronous mode, using the interrupt mode. The peripheral initialization is done using the LL initialization function to demonstrate LL init usage.	-	MX
		USART_Communication_Tx_IT_Init	How to configure GPIO and USART peripherals to send characters asynchronously to a HyperTerminal (PC) in interrupt mode. This example is based on the STM32U3xx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	MX
		USART_Communication_Tx_IT_VCP_Init	How to configure GPIO and USART peripherals to send characters asynchronously to a HyperTerminal (PC) in interrupt mode. This example is based on the STM32U3xx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	MX
		USART_Communication_Tx_Init	How to configure GPIO and USART peripherals to send characters asynchronously to a HyperTerminal (PC) in polling mode. If the transfer cannot be completed within the allocated time, a timeout allows exiting from the sequence with a timeout error code. This example is based on the STM32U3xx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	MX
		USART_Communication_Tx_VCP_Init	How to configure GPIO and USART peripherals to send characters asynchronously to a HyperTerminal (PC) in polling mode. If the transfer cannot be completed within the allocated time, a timeout allows exiting from the sequence with a timeout error code. This example is based on the STM32U3xx USART LL API. The peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	MX
	UTILS	UTILS_ConfigureSystemClock	How to use the UTILS LL API to configure the system clock using the MSIS as a source clock.	-	MX
		UTILS_ReadDeviceInfo	How to read the UID, Device ID and Revision ID, and save them into a global information buffer.	-	MX
	WWDG	WWDG_RefreshUntilUserEvent_Init	How to configure the WWDG to periodically update the counter and generate an MCU WWDG reset when a user button is pressed. The peripheral initialization uses the LL unitary service functions for optimization purposes (performance and size).	-	MX
	Total number of examples_ll			**0**	**66**
Examples_MIX	ADC	ADC_SingleConversion_TriggerSW_IT	How to use the ADC to convert a single channel at each software start. The conversion is performed using the interrupt programming model.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Examples_MIX	CRC	CRC_PolynomialUpdate	How to use the CRC peripheral through the STM32U3xx 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 STM32U3xx DMA HAL and LL APIs. The LL API is used for performance improvement.	-	MX
	SPI	SPI_FullDuplex_ComPolling_Master	How to handle data buffer transmission/reception between two boards via SPI using the polling mode.	-	MX
		SPI_FullDuplex_ComPolling_Slave	How to handle data buffer transmission/reception between two boards via SPI using the polling mode.	-	MX
		SPI_HalfDuplex_ComPollingIT_Master	How to handle data buffer transmission/reception between two boards via SPI using the polling (LL driver) and interrupt modes (HAL driver).	-	MX
		SPI_HalfDuplex_ComPollingIT_Slave	How to handle data buffer transmission/reception between two boards via SPI using the polling (LL driver) and interrupt modes (HAL driver).	-	MX
	TIM	TIM_PWMInput	How to use the TIM to measure an external signal frequency and duty cycle.	-	MX
	Total number of examples_mix			**0**	**8**
Applications	-	OpenBootloader	This application exploits OpenBootloader middleware to demonstrate how to develop an IAP application and use it.	-	X
	FileX	Fx_File_Edit_Standalone	This application provides an example of FileX stack usage on the NUCLEO-U385RG-Q board, running in standalone mode (without ThreadX). It demonstrates how to create a Fat File system on the internal SRAM using FileX.	-	MX
	MbedTLS_HW_ALT	Cipher_AES_CBC_EncryptDecrypt_HAL	How to use the PSA reference API to perform encryption and decryption using the AES CBC algorithm.	-	X
		Cipher_AES_CCM_Encrypt_Decrypt_HAL	How to use the PSA reference API to perform authenticated encryption and verified decryption using the AES CCM algorithm.	-	X
		Cipher_AES_GCM_Encrypt_Decrypt_HAL	How to use the PSA reference API to perform authenticated encryption and verified decryption using the AES GCM algorithm.	-	X
		ECC_ECDH_SharedSecretGeneration_HAL	How to use the cryptographic reference API to establish a shared secret using the ECDH algorithm over the SECP256 curve.	-	X
		ECC_ECDSA_SignVerify_HAL	How to use the PSA reference API to sign and verify a message using the ECDSA algorithm over the SECP256 curve.	-	X
		Encrypted_ITS_KeyImport	How to use the PSA ITS alternative encrypted implementation to import the AES-CBC key and store it in user-persistent storage. The key is encrypted by the ITS.	-	X
		Hash_SHA2_Digest_HAL	How to use the PSA reference API to digest a message using the SHA256 algorithm.	-	X

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Applications	MbedTLS_HW_ALT	MAC_HMAC_SHA2_AuthenticateVerify_HAL	How to use the PSA reference API to authenticate and verify a message using the HMAC SHA256 algorithm.	-	X
		RSA_PKCS1v1.5_SignVerifyCRT_HAL	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v1.5 compliant algorithm.	-	X
		RSA_PKCS1v1.5_SignVerify_HAL	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v1.5 compliant algorithm.	-	X
		RSA_PKCS1v2.2_EncryptDecryptCRT_HAL	How to use the PSA reference API to encrypt and decrypt a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
		RSA_PKCS1v2.2_EncryptDecrypt_HAL	How to use the PSA reference API to encrypt and decrypt a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
		RSA_PKCS1v2.2_SignVerifyCRT_HAL	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
		RSA_PKCS1v2.2_SignVerify_HAL	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
	MbedTLS_HW_KWE	AES_WrapKey_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap AES-GCM, AES-ECB, and AES-CBC keys.	-	X
		Cipher_AES_CBC_EncryptDecrypt_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap the AES-CBC private key and use the wrapped key to perform encryption and verify the decryption using the AES CBC algorithm.	-	X
		Cipher_AES_CCM_Encrypt_Decrypt_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap the AES-CCM private key and use the wrapped key to perform authenticated encryption and verify the decryption using the AES CCM algorithm.	-	X
		Cipher_AES_ECB_EncryptDecrypt_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap the AES-ECB private key and use the wrapped key to perform encryption and verify the decryption using the AES CBC algorithm.	-	X
		Cipher_AES_GCM_Encrypt_Decrypt_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap the AES-GCM private key and use the wrapped key to perform authenticated encryption and verified the decryption using the AES GCM algorithm.	-	X
		ECC_ECDH_GenerateWrappedKey_KWE	How to use the PSA Crypto opaque driver based on the STM32 Key Wrap Engine to generate the ECDH private key.	-	X
		ECC_ECDH_SharedSecretGeneration_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap the ECDH private key and use the wrapped key to generate the shared secret for ECDH key agreement.	-	X
		ECC_ECDH_WrapKey_KWE	How to use the PSA Crypto opaque driver based on the STM32 Key Wrap Engine to wrap the ECDH private key.	-	X
		ECC_ECDSA_ExportPublicKey_KWE	How to use the PSA reference API to sign and verify a message using the ECDSA algorithm over the SECP256 curve.	-	X

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Applications	MbedTLS_HW_KWE	ECC_ECDSA_GenerateWrappedKey_KWE	How to use the PSA Crypto opaque driver based on the STM32 Key Wrap Engine to generate the ECDSA private key.	-	X
		ECC_ECDSA_Sign_KWE	How to use the PSA Crypto opaque driver based on the STM32 Key Wrap Engine to wrap the ECDSA private key and use the wrapped key for ECDSA signature.	-	X
		ECC_ECDSA_WrapKey_KWE	How to use the PSA Crypto opaque driver based on the STM32 Key Wrap Engine to wrap the ECDSA private key.	-	X
		Initial_Attestation_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to sign an entity attestation token (ETA) for initial attestation using the device unique authentication key (DUA) in wrapped form.	-	X
		RSA_PKCS1v1.5_Sign_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap the RSA private key and use the wrapped key to compute RSA PCKS1v1.5 signature.	-	X
		RSA_PKCS1v2.2_Decrypt_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to warp the RSA private key and use the wrapped key for RSA PCKS1v2.2 decryption.	-	X
		RSA_PKCS1v2.2_Sign_KWE	How to use the PSA Crypto opaque driver based on STM32 Key Wrap Engine to wrap the RSA private key and use the wrapped key to compute RSA PCKS1v2.2 signature.	-	X
		RSA_WrapKey_KWE	How to use the PSA Crypto opaque driver based on the STM32 Key Wrap Engine to wrap user RSA key.	-	X
	MbedTLS_SW	Cipher_AES_CBC_EncryptDecrypt_MBED	How to use the PSA reference API to perform encryption and decryption using the AES CBC algorithm.	-	X
		Cipher_AES_CCM_Encrypt_Decrypt_MBED	How to use the PSA reference API to perform authenticated encryption and verified decryption using the AES CCM algorithm.	-	X
		Cipher_AES_GCM_Encrypt_Decrypt_MBED	How to use the PSA reference API to perform authenticated encryption and verified decryption using the AES GCM algorithm.	-	X
		Cipher_ChachaPoly_AuthEnc_VerifDec_MBED	How to use the PSA reference API to perform authenticated encryption and verified decryption using the Chacha-Poly1305 algorithm.	-	X
		DRBG_RandomGeneration_MBED	How to use the PSA reference API to generate random numbers using the DRBG module.	-	X
		ECC_ECDH_SharedSecretGeneration_MBED	How to use the PSA reference API to establish a shared secret using the ECDH algorithm over the SECP256 curve.	-	X
		ECC_ECDSA_SignVerify_MBED	How to use the PSA reference API to sign and verify a message using the ECDSA algorithm over the SECP256 curve.	-	X
		Hash_SHA2_Digest_MBED	How to use the PSA reference API to digest a message using the SHA256 algorithm.	-	X
		MAC_AES_CMAC_AuthenticateVerify_MBED	How to use the PSA reference API to authenticate and verify a message using the AES CMAC algorithm.	-	X

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Applications	MbedTLS_SW	MAC_HMAC_SHA2_AuthenticateVerify_MBED	How to use the PSA reference API to authenticate and verify a message using the HMAC SHA256 algorithm.	-	X
		RSA_PKCS1v1.5_SignVerifyCRT_MBED	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v1.5 compliant algorithm.	-	X
		RSA_PKCS1v1.5_SignVerify_MBED	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v1.5 compliant algorithm.	-	X
		RSA_PKCS1v2.2_EncryptDecryptCRT_MBED	How to use the PSA reference API to encrypt and decrypt a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
		RSA_PKCS1v2.2_EncryptDecrypt_MBED	How to use the PSA reference API to encrypt and decrypt a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
		RSA_PKCS1v2.2_SignVerifyCRT_MBED	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
		RSA_PKCS1v2.2_SignVerify_MBED	How to use the PSA reference API to sign and verify a message using the RSA PKCS#1 v2.2 compliant algorithm.	-	X
	ROT	OEMiROT_Appli	This project provides a OEMiROT boot path full secure application example. The boot is performed through the OEMiROT boot path after the authenticity and the integrity checks of the project firmware and project data images.	-	X
		OEMiROT_Appli_TrustZone	This project provides a OEMiROT boot path application example. The bot is performed through the OEMiROT boot path after the authenticity and the integrity checks of the project firmware and project data images.	-	X
		OEMiROT_Boot	This project provides an OEMiROT example. The OEMiROT boot path performs the authenticity and the integrity checks of the project firmware and data images.	-	X
	ThreadX	Tx_LowPower	This application provides an example of Azure® RTOS ThreadX stack usage. It shows how to develop an application using the ThreadX® low-power feature.	-	MX
		Tx_MPU	This application provides an example of Azure® RTOS ThreadX stack usage. It shows how to develop an application using the ThreadX® module feature.	-	X
		Tx_Thread_Creation	This application provides an example of Azure® RTOS ThreadX stack usage. It shows how to develop an application using the ThreadX® thread management APIs.	-	MX
		Tx_Thread_MsgQueue	This application provides an example of Azure® RTOS ThreadX stack usage. It shows how to develop an application using the ThreadX® message queue APIs.	-	MX
		Tx_Thread_Sync	This application provides an example of Azure® RTOS ThreadX stack usage. It shows how to develop an application using the ThreadX® synchronization APIs.	-	MX
	USBX	Ux_Device_CDC_ACM	This application provides an example of Azure® RTOS USBX stack usage on the NUCLEO-U385RG-Q board. It shows how to develop a USB Device communication Class (CDC_ACM) based application.	-	MX

Level	Module name	Project name	Description	STM32U3_CUSTOM_HW[1]	NUCLEO-U385RG-Q
Applications	USBX	Ux_Device_HID_Standalone	This application provides an example of USBX stack usage on the NUCLEO-U385RG-Q board. It shows how to develop a USB Device Human Interface "HID" mouse based bare metal application.	-	MX
		Ux_Host_HID	This application provides an example of Azure® RTOS USBX stack usage.	-	MX
		Ux_Host_HID_Standalone	This application provides an example of Azure® USBX stack usage.	-	MX
	Total number of applications			**0**	**61**
Total number of projects				**5**	**325**

[1] *The firmware examples listed in this column are not supported by the NUCLEO-U385RG-Q board. They can be used with a custom board.*

2 Reference documents

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

- Latest release of the STM32U3 firmware package
- *Getting started with STM32CubeU3 for STM32U3 series* (UM3427)
- *Description of STM32U3xx HAL and low-layer drivers* (UM3439)

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

Revision history

Table 2. Document revision history

Date	Version	Changes
19-Feb-2025	1	Initial release.

Contents

List of tables