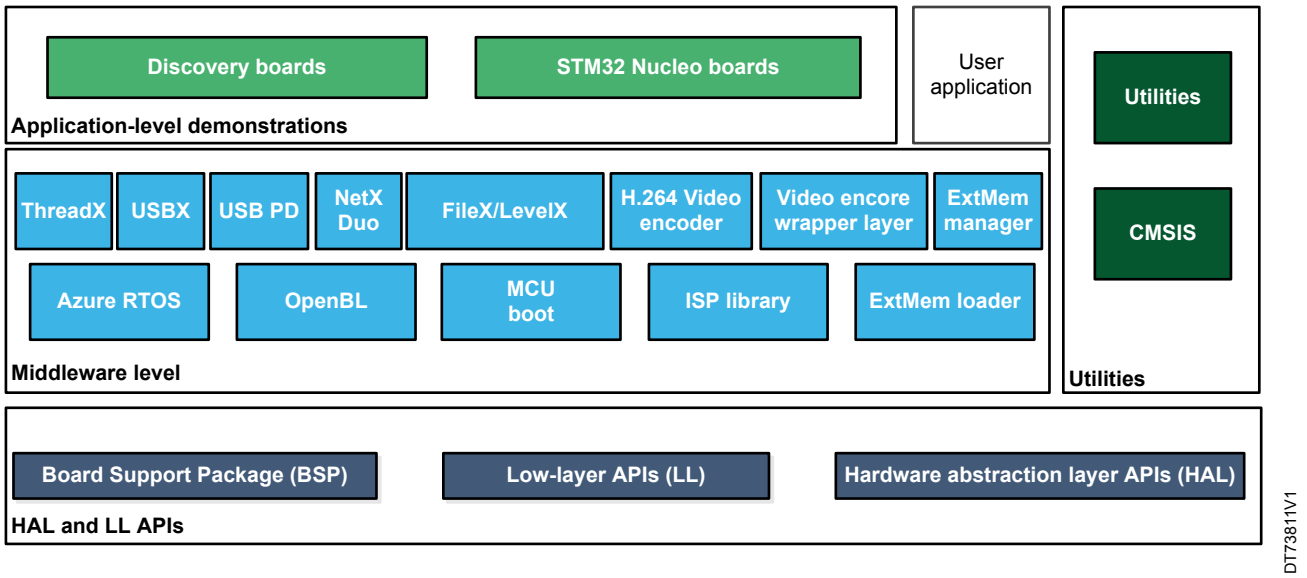
STM32Cube MCU package examples for STM32N6 MCUs

Introduction

The STM32CubeN6 STM32Cube MCU Package is delivered with a rich set of examples running on STMicroelectronics boards. The examples are organized by boards. They are provided with preconfigured projects for the main supported toolchains (refer to Figure 1). In the STM32CubeN6 MCU Package, most examples and application projects are generated with the STM32CubeMX tool (starting from version v6.13.0) to initialize the system, peripherals, and middleware stacks. The user can open the provided .ioc file when available in STM32CubeMX to modify the settings, and add additional peripherals, middleware components, or both, to build their final application. For more information about STM32CubeMX, refer to the STM32CubeMX for STM32 configuration and initialization C code generation user manual (UM1718).

Figure 1. STM32CubeN6 firmware components



1 Reference documents

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

Table 1. List of items

Reference	Name/address	Title
[1]	-	The latest release of the STM32CubeN6 MCU Package for the 32-bit microcontrollers in the STM32N6xx devices based on the Arm® Cortex®-M processor
[2]	UM3249	Getting started with STM32CubeN6 for STM32N6 MCUs
[3]	UM3245	Description of STM32N6xx HAL and low-layer drivers
[4]	UM1718	STM32CubeMX for STM32 configuration and initialization C code generation

This document applies to STM32N6 series Arm®Cortex®-M55-based microcontrollers.

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

2 STM32CubeN6 examples

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

- **Templates**
  Template projects are provided to enable the user to quickly build a firmware application using HAL and BSP drivers on a given board. Different kinds of templates are provided.
- **Template_LL**
  After boot ROM execution, the template runs from the internal SRAM based on low-layer drivers.
- **Template**
  After boot ROM execution, the template runs from the internal SRAM based on HAL drivers.
- **Template_FSBL_LRUN**
  After boot ROM execution, FSBL (first stage boot loader) runs from the internal SRAM, followed by the application.
  Template_FSBL_LRUN project is composed of two sub-projects:
  - **FSBL:** FSBL execution from the internal RAM with jump to application in internal RAM.
  - **Appli:** application execution from the internal RAM.
- **Template_FSBL_XIP**
  After boot ROM execution, FSBL runs from the internal SRAM, followed by the application running from the external flash memory.
  - Template_FSBL_XIP project is composed of three sub-projects:
  - **FSBL:** FSBL execution from the internal RAM with jump to application in external flash memory.
  - **Appli:** application execution from external flash memory.
  - **ExtMemLoader:** build the external flash memory loader for the board.
- **Template_Isolation_LRUN**
  After boot ROM execution, FSBL runs from internal SRAM, followed by secure application then nonsecure application.
  Template_Isolation_LRUN project is composed of three sub-projects:
  - **FSBL:** FSBL execution from the internal RAM with jump to secure application in the internal RAM.
  - **AppliSecure:** secure application execution from the internal RAM with jump to nonsecure application in the internal RAM.
  - **AppliNonSecure:** nonsecure application execution from the internal RAM.
- **Template_Isolation_XIP**
  After boot ROM execution, FSBL runs from internal SRAM, followed by secure application then nonsecure application both running from the external flash memory
  Template_Isolation_XIP project is composed of four sub-projects:
  - **FSBL:** FSBL execution from internal RAM with jump to secure application in the external flash memory.
  - **AppliSecure:** secure application execution from external flash memory with jump to nonsecure application in external flash memory.
  - **AppliNonSecure:** nonsecure application execution from the external flash memory.
  - **ExtMemLoader:** build the external flash memory loader for the board.
- **Template_ROT**
  OEMuRoT boot path after authenticity and integrity checks of the project firmware and project data images.
- **Templates_Board**
  After boot ROM execution, the template runs from the internal SRAM. STM32CubeMX "Start my project from ST Board" provides an easy access to ST boards' features with demonstration code based on HAL and BSP drivers.

Note: The examples below follow the software architecture described for the templates here-above. Most of the examples are based on Template and on Template_FSBL_LRUN.

- **Examples**
  These examples use only the HAL and BSP drivers (middleware components are 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 the RIF peripheral to ensure the proper memory protection or peripheral access limitation. 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 are 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 in order to combine the advantages of both APIs:
  – HAL offers high-level function-oriented APIs with a 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 by middleware (one folder per middleware, such as USB host). The integration of applications that use several middleware stacks is also supported.

The examples are located under `STM32Cube_FW_N6_VX.Y.Z\Projects\`.

All the examples have the same structure and can contain up to four contexts or subprojects:

- *\FSBL
  *\Inc folder containing all the header files for the FSBL part.
  *\Src folder containing the code sources for the FSBL part.
- • *\Appli
  *\Inc folder containing all the header files for the user application part.
  *\Src folder containing all the code sources for the user application part.
- *\AppliSecure
  *\Inc folder containing all the header files for the secure user application part.
  *\Src folder containing all the code sources for the "secure user application" part.
- *\AppliNonSecure
  *\Inc folder containing all the header files for the nonsecure user application part.
  *\Src folder containing all the code sources for the nonsecure user application part.
  *\EWARM, *\MDK-ARM and *\STM32CubeIDE folders, containing the preconfigured project for each toolchain.
  *\readme.html and *\README.md describing the behavior of the example, and the environment required to run the example.
  *.ioc file that allows users to open firmware examples within STM32CubeMX.
- \ExtMemloader folder containing files used to generate a binary able to download an application to an external memory.

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 2 contains the list of examples provided with the STM32CubeN6 MCU Package.

In this table, the label MX means that 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 labelled with X are manually created to demonstrate the product features. Read the project *\README.md and *\readme.html files for user option bytes configuration.

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

Table 2. STM32CubeN6 firmware examples

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Templates_Board	-	Starter project	This project provides a reference template for the STM32N6570-DK board based on the STM32Cube HAL API that can be used to build any firmware application. Note that security is enabled by default on Cortex-M55.	-	MX	-
	Total number of templates_board: 1			0	1	0
Templates	-	Template	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application. Note that security is enabled by default on Cortex-M55.	-	MX	MX
		Template_FSBL_LRUN	This project provides a reference FSBL LRUN template based on the STM32Cube HAL API that can be used to build any firmware application to execute in internal RAM (sub-project Appli).	-	X	X
		Template_FSBL_XIP	This project provides a reference FSBL XIP template based on the STM32Cube HAL API that can be used to build any firmware application to execute in external Flash (sub-project Appli).	-	X	X
		Template_Isolation_LRUN	This project provides a reference TrustZone Isolation LRUN template based on the STM32Cube HAL API that can be used to build any firmware application made of a secure binary and of a non-secure binary to execute in internal SRAM.	-	X	X
		Template_Isolation_XIP	This project provides a reference TrustZone Isolation XIP template based on the STM32Cube HAL API that can be used to build any firmware application made of a secure binary and of a non-secure binary to execute in external flash memory.	-	X	X
	Total number of templates: 10			0	5	5
ROT_Provisioning	-	BootROM	This section provides the necessary scripts to generate and program boot ROM keys.	-	X	X
		OEMuRoT	This section provides available configuration scripts for OEMuRoT example.	-	X	X
	Total number of templates_board: 4			0	2	2

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Templates_ROT	-	-	This project provides a OEMuRoT boot path application example. Boot is performed through OEMuRoT boot path after authenticity and integrity checks of the project firmware and project data images.	-	X	-
	Total number of templates_board: 1			0	1	0
Templates_LL	-	Template	This project provides a reference template based on the STM32Cube LL API that can be used to build any firmware application. Note that security is enabled by default on Cortex-M55.	-	MX	MX
	Total number of templates_ll: 2			0	1	1
Examples	-	BSP	This example provides a description of how to use the different BSP drivers.	-	X	-
	ADC	ADC_AnalogWatchdog	How to use an ADC peripheral with an ADC analog watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds.	-	-	MX
		ADC_FixedTriggerLatency	How to use an ADC peripheral to perform a single ADC conversion on a channel at each trigger event from a timer without any uncertainty (fixed trigger latency).	-	-	MX
		ADC_MultiChannel SingleConversion	How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence.	-	-	MX
		ADC_SingleConversion_ TriggerTimer_DMA	How to use an ADC peripheral to perform a single ADC conversion on a channel at each trigger event from a timer. Converted data is transferred by DMA into a table in RAM memory.	-	-	MX
	BSEC	BSEC_OTP_Config	How to configure OTP (One-Time Programmable) bits.	-	MX	-
		BSEC_ShadowRegisters	This project is targeted to run on STM32N6 device on NUCLEO-N657X0 board from STMicroelectronics.	-	-	MX
	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
		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
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	-	-	MX
		CORTEX_CACHE	This project provides a CORTEXM cache example based on the CMSIS API that can be used to build any firmware application to execute from internal flash memory.	-	-	MX

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6 570-DK	NUCLEO-N657X0-Q
Examples	CORTEX	CORTEX_HELIUM	This project describes how to use the SIMD Helium instructions and the performance gained.	-	-	MX
		CORTEX_InterruptSwitch_TrustZone	This project describes how to switch an interrupt from secure to nonsecure domain.	-	-	X
		CORTEX_MPU	This project provides a CORTEXM cache example based on the CMSIS API that can be used to build any firmware application to execute from internal flash memory.	-	-	MX
	CRC	CRC_ReverseModes	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), input or output data reversal features are enabled, using a fixed generator polynomial used in MPEG2: $(0x4C11DB7) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$ (1)	-	-	MX
	CRYP	CRYP_AES_GCM	How to use the CRYPTO peripheral to encrypt and decrypt data using AES with Galois/Counter mode (GCM).	-	-	MX
		CRYP_SAES_SharedKey	How to use the Secure AES co-processor (SAES) peripheral to share application keys with AES peripheral.	-	-	MX
		CRYP_SAES_WrapKey	How to use the Secure AES co-processor (SAES) peripheral to wrap application keys using hardware secret key DHUK then use it to encrypt in polling mode.	-	-	MX
	DMA	DMA_LinkedList	This example describes 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 reload the next transfer parameters, and starts it (without CPU intervention).	-	-	MX
		DMA_RAMToRAM	This example describes how to use DMA to transfer a word data buffer from SRAM memory to embedded SRAM through the HAL API. Isolation of the HPDMA channel is aligned with targeted memories.	-	-	MX
		DMA_RIF_management	How to use CID information to isolate DMA channels.	-	-	MX
		DMA_Trigger	How to configure and use the DMA HAL API to perform DMA triggered transactions.	-	-	MX

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples	DMA2D	DMA2D_MemToMemWithPFC	This example provides a description of how to configure the DMA2D peripheral in memory-to-memory transfer mode with pixel format conversion (PFC) mode.	-	MX	-
		DMA2D_MemoryToMemory	This example provides a description of how to configure DMA2D peripheral in memory-to-memory transfer mode.	-	MX	-
	DTS	DTS_GetTemperature	How to configure and use the DTS to get the temperature of the die.	-	-	MX
	FDCAN	FDCAN_Loopback	This example describes how to configure the FDCAN peripheral to operate in loopback mode.	-	-	MX
	FMC	FMC_NAND_Load_And_Run	This example describes how to configure the FMC controller to access the NAND memory.	MX	-	-
		FMC_SDRAM_DataMemory	This example describes how to configure the FMC controller to access the SDRAM memory.	MX	-	-
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	-	-	MX
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	-	MX	MX
	HAL	HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as main source of time base instead of Systick.	-	-	MX
	HASH	HASH_HMAC_SHA2_384	This example provides a short description of how to use the HASH peripheral to hash data using long key and SHA_384 algorithm.	-	-	MX
		HASH_SHA224SHA256_DMA	How to use the HASH peripheral to hash data with SHA224 and SHA256 algorithms.	-	-	MX
	I2C	I2C_Sensor_Private_Command_IT	How to handle I2C data buffer transmission/reception between STM32N6xx Nucleo and X-NUCLEO-IKS4A1 using an interrupt.	-	-	MX
		I2C_TwoBoards_AdvComIT	How to handle I2C data buffer transmission/reception between two boards via DMA.	-	-	MX
		I2C_TwoBoards_ComDMA	How to handle I2C data buffer transmission/reception between two boards, via DMA.			
		I2C_TwoBoards_ComPolling	How to handle I2C data buffer transmission/reception between two boards in polling mode.	-	-	MX
		I2C_TwoBoards_MultiMasterIT_Master	How to handle I2C data buffer communication between two boards, using an interrupt and two masters and one slave.	-	-	MX
		I2C_TwoBoards_MultiMasterIT_Slave	How to handle I2C data buffer communication between two boards, using an interrupt and two masters and one slave.	-	-	MX

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples	I2C	I2C_TwoBoards_RestartAdvComIT	How to perform multiple I2C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	-	-	MX
	I3C	I3C_Controller_Direct_Command_DMA	How to handle a direct command procedure between an I3C controller and an I3C target using DMA.	-	-	MX
		I3C_Sensor_Direct_Command_DMA	How to handle a direct command procedure between STM32N6 Nucleo and X-NUCLEO-IKS01A3 using DMA.	-	-	MX
		I3C_Sensor_Private_Command_IT	How to handle I3C as controller data buffer transmission/reception between STM32N6 Nucleo and X-NUCLEO-IKS01A3 using interrupt.	-	-	MX
		I3C_Target_Direct_Command_DMA	How to handle a direct command procedure between an I3C controller and an I3C target using controller in DMA.	-	-	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
	JPEG	JPEG_DecodingFromOSPI_DMA	This project demonstrates how to decode a JPEG image using using the JPEG HW decoder in DMA mode.	-	X	-
		JPEG_EncodingFromOSPI_DMA	This project demonstrates how to encode an RGB image using using the JPEG HW encoder in DMA mode.	-	X	-
	LPTIM	LPTIM_PulseCounter	How to configure and use, through the LPTIM HAL API, the LPTIM peripheral to count pulses.	-	-	MX
		LPTIM_Timeout	How to implement, through the HAL LPTIM API, a timeout with the LPTIMER peripheral, to wake up the system from a low-power mode.	-	-	MX
	LPUART	LPUART_WakeUpFromStop	Configuration of an LPUART to wake up the MCU from Stop mode when a given stimulus is received.	-	-	MX
	LTDC	LTDC_Horizontal_Mirroring	This example shows how to use the LTDC IP to Mirror the displayed image and it is based on the STM32Cube HAL API that can be used to build any firmware application. Note that security is enabled by default on Cortex-M55.	-	MX	-
		LTDC_YUV_Full_Planar	This example shows how to use the LTDC IP to display an YUV 420 full planar image and it is based on the STM32Cube HAL API that can be used to build any firmware application. Note that security is enabled by default on Cortex-M55.	-	MX	-
	MCE	MCE_AESEncryptDecryptData_Block_AES_256	This project provides a description of how encrypt and decrypt data from external memory (PSRAM).	-	X	-
		MCE_AESEncryptDecryptData_Stream_AES_128	This project provides a description of how encrypt and decrypt data from external memory (PSRAM).	-	X	-

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples	MCE	MCE_ExecuteAESCryptedCode_Stream_AES_128	This project provides a description of how to run encrypted application code from external flash memory. The decryption is performed on the fly using the MCE IP (AES128 encryption algorithm).	-	X	-
		MCE_NoekeonEncryptDecryptData	This project provides a description of how to encrypt and decrypt data from external memory (PSRAM).	-	MX	-
	MDF	MDF_AudioRecorder	This project demonstrates how to configure MDF to perform PCM record in DMA circular mode.	-	MX	-
	PKA	PKA_ECCscalar Multiplication	How to use the PKA peripheral to execute ECC scalar multiplication. This allows generating a public key from a private key.	-	-	MX
		PKA_ECDSA_Verify	How to determine if a given signature is valid regarding the Elliptic curve digital signature algorithm (ECDSA).	-	-	MX
	PSSI	PSSI_Master_Single_Com	How to handle a single communication procedure using two boards with PSSI in polling mode.	-	-	MX
		PSSI_Slave_Single_Com	How to handle a single communication procedure using two boards with PSSI in polling mode.	-	-	MX
	PWR	PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	-	-	MX
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using a wake-up pin.	-	X	X
		PWR_STANDBY_RTC	How to enter the Standby mode and wake-up from this mode by using the RTC wake-up timer.	-	X	X
		PWR_STANDBY_TrustZone	How to enter the Standby mode with a joint secure/nonsecure application and wake up from this mode by using the RTC wake-up timer.	-	X	-
		PWR_STOP	How to enter the Stop mode and wake up from this mode by using an interrupt.	-	-	MX
		PWR_STOP_RTC	How to enter the Stop mode and wake up from this mode by using an external reset or the RTC wake-up timer.	-	-	MX
	RAM CFG	RAMCFG_ECC_Error_Generation	How to configure and use the RAMCFG HAL API to manage ECC errors via RAMCFG peripheral.	-	-	MX
	RCC	RCC_ClockConfig	Configuration of the system clock (SYSCLK) and modification of the clock settings in Run mode using the RCC HAL API.	-	MX	MX
		RCC_LSEConfig	Enabling/disabling of the low-speed external (LSE) RC oscillator (about 32 KHz) at run time using the RCC HAL API.	-	-	MX

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples	RCC	RCC_SwitchClock	This example describes how to use the RCC HAL API to configure the CPU and system buses clocks and modify the clock settings on run time.	-	MX	MX
	RIF	RIF_Memory	This example describes how to use the RIF HAL API to configure a RISAF in order to protect a memory from illegal accesses.	-	-	MX
		RIF_Peripheral	This example describes how to use the RIF HAL API to configure a RISAF in order to protect a memory from illegal accesses.	-	-	X
	RNG	RNG_Config	How to configure the RNG peripheral with HAL API.	-	-	MX
		RNG_MultiRNG	How to generate random numbers with the RNG HAL API.	-	-	MX
		RNG_MultiRNG_IT	How to generate random numbers under interrupt with the RNG HAL API.	-	-	MX
	RTC	RTC_ActiveTamper	Configuration of the active tamper detection with backup registers erase.	-	-	MX
		RTC_Alarm	Configuration and generation of an RTC alarm using the RTC HAL API.	-	-	MX
		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
	SD	SD_ReadWrite_DMA	This example performs some write and read transfers to SD Card with SDMMC IP internal DMA mode. Note that security is enabled by default on Cortex-M55.	-	X	-
	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_CrcComIT_Master	Data buffer transmission/reception with CRC between two boards via SPI using IT.	-	-	MX
		SPI_FullDuplex_CrcComIT_Slave	Data buffer transmission/reception with CRC between two boards via SPI using IT.	-	-	MX
	SYSCFG	FLEXMEM_Configurations	This project is targeted to run on STM32N6 devices on NUCLEO-N657X0 board from STMicroelectronics.	-	-	MX
		SYSCFG_WritePostingErrors	This example shows how to detect write posting errors.	-	-	MX

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples	TIM	TIM_DMA	Use of the DMA with TIMER update request to transfer data from memory to TIMER capture compare register 3 (TIMx_CCR3).	-	-	MX
		TIM_DeadtimeInsertion	This example shows how to configure the TIM peripheral in PWM (Pulse Width Modulation) mode with asymmetric deadtime insertion between the complementaty channels.	-	-	MX
		TIM_Encoder	This example shows how to configure the TIM1 peripheral in encoder mode to determinate the rotation direction.	-	-	MX
		TIM_ExtTriggerSynchro	This example shows how to synchronize TIM peripherals in cascade mode with an external trigger.	-	-	MX
		TIM_ExternalClockMode1	This example shows how to configure the TIM peripheral in external clock mode 1 and use the button as a clock source to light a LED after five presses.	-	-	MX
		TIM_PWMInput	How to use the TIM peripheral to measure the frequency and duty cycle of an external signal.	-	-	MX
		TIM_PWMOutput	This example shows how to configure the TIM peripheral in PWM (Pulse Width Modulation) mode.	-	-	MX
		TIM_TimeBase	This example shows how to configure the TIM peripheral to generate a time base of one second with the corresponding Interrupt request.	-	-	MX
	UART	UART_Console	How to use the HAL UART API for UART transmission (printf/getchar) via console with user interaction.	-	-	MX
		UART_HyperTerminal_DMA	UART transmission (transmit/receive) in DMA mode between a board and an HyperTerminal PC application.	-	-	MX
		UART_HyperTerminal_IT	UART transmission (transmit/receive) in Interrupt mode between a board and an 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 reception to IDLE event in circular DMA mode.	-	-	MX
		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
	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

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples	XSPI	XSPIM_SwappedMode	This project provides a description of how to configure XSPIM IO manager peripheral and communicate with external memories in Swapped mode.	-	MX	-
		XSPI_HyperFLASH_ReadWrite	How to use an XSPI NOR memory in automatic polling mode.	MX	-	-
		XSPI_NOR_AutoPolling_DTR	How to use a XSPI NOR memory in memory-mapped mode.	-	MX	MX
		XSPI_NOR_Memory Mapped_DTR	How to use a XSPI NOR memory in memory-mapped mode.	-	MX	MX
		XSPI_PSRAM_Memory Mapped	How to use an XSPI PSRAM memory in memory-mapped mode.	-	MX	-
	Total number of examples: 116			3	25	88
Examples_LL	CORTEX	CORTEX_MPU	Presentation of the MPU features. This example configures 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
	EXTI	EXTI_ToggleLedOnIT_Init	This example describes how to configure the EXTI and use GPIOs to toggle the user LEDs available on the board when a user button is pressed. This example is based on the STM32N6 LL API. Peripheral initialization is done using 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 STM32N6 LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	-	-	MX
	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_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_WakeUpFromStop_IT_Init	How to handle the reception of a data byte from an I2C slave device in Stop 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

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
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
	RCC	RCC_OutputSystemClockOnMCO	Configuration of MCO pin (PC9) to output the system clock.	-	-	MX
		RCC_UseHSEasSystemClock	Use of the RCC LL API to start the HSE and use it as system clock.	-	-	MX
		RCC_UseHSI_PLLasSystemClock	Modification of the PLL parameters in run time.	-	-	MX
	RNG	RNG_GenerateRandomNumbers	Configuration of the RNG to generate 32-bit long random numbers. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	RTC	RTC_Alarm_Init	Configuration of the RTC LL API to configure and generate an alarm using the RTC peripheral. The peripheral initialization uses the LL initialization function.	-	-	MX
		RTC_TimeStamp_Init	Configuration of the Timestamp using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	SPI	SPI_OneBoard_HalfDuplex_DMA_Init	Configuration of 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 STM32N6xx SPI LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	-	MX
		SPI_TwoBoards_FullDuplex_DMA_Master_Init	Data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32N6xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		SPI_TwoBoards_FullDuplex_DMA_Slave_Init	Data buffer transmission and reception via SPI using DMA mode. This example is based on the STM32N6xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	TIM	TIM_BreakAndDeadtime_Init	Configuration of the TIM peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined deadtime value, use the break feature, and lock the break and dead-time configuration.	-	-	MX

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples_LL	TIM	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 STM32N6xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		TIM_InputCapture_Init	Use of the TIM peripheral to measure a periodic signal frequency provided either by an external signal generator or by another timer instance. This example is based on the STM32N6xx 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 STM32N6xx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL Init.	-	-	MX
		TIM_OutputCompare_Init	Configuration of the TIM peripheral to generate an output waveform in different output compare modes. This example is based on the STM32N6xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		TIM_PWMOutput_Init	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle.	-	-	MX
		TIM_TimeBase_Init	Configuration of the TIM peripheral to generate a timebase. This example is based on the STM32N6 TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	USART	USART_Communication_Rx_IT_Continuous_VCP_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 purpose (performance and size).	-	-	MX
		USART_Communication_Rx_IT_VCP_Init	This example shows how to configure GPIO and USART peripheral for receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL initialization function to demonstrate LL init usage.	-	-	MX

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Examples_LL	USART	USART_Communication_Tx_IT_VCP_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 STM32N6xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	MX
		USART_Communication_Tx_VCP_Init	This example shows how to configure GPIO and USART peripherals to send characters asynchronously to an HyperTerminal (PC) in Polling mode. If the transfer could not be completed within the allocated time, a timeout allows to exit from the sequence with a Timeout error code. This example is based on STM32N6xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	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, Device ID and 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 the LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	Total number of examples_ll: 30			0	0	30
Examples_MIX	UART	UART_HyperTerminal_IT	Use of a UART to transmit data (transmit/receive) between a board and an HyperTerminal PC application in Interrupt mode. This example describes how to use the USART peripheral through the STM32N6 UART HAL and LL API, the LL API being used for performance improvement.	-	-	MX
	Total number of examples_mix: 1			0	0	1
Applications	-	OpenBootloader	This application exploits OpenBootloader Middleware to demonstrate how to develop an application that can be used to program OTP words or external memories.	-	X	-
	DCMIPP	DCMIPP_ContinuousMode	This example shows how to use the DCMIPP IP in continuous mode as a camera serial interface and it is based on the STM32Cube HAL API that can be used to build any firmware application. Note that security is enabled by default on Cortex-M55.	-	X	-

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Applications	DCMIPP	DCMIPP_SnapshotDecimationMode	This example shows how to use the DCMIPP IP in Snapshot Mode as a camera serial interface and it is based on the STM32Cube HAL API that can be used to build any firmware application. Note that security is enabled by default on Cortex-M55.	-	X	-
	FileX	Fx_File_Edit_Standalone	This application provides an example of Azure® RTOS FileX stack usage on NUCLEO-N657X0-Q board, running in standalone mode (without ThreadX). It demonstrates how to create a fat file system on the internal SRAM memory using FileX API.	-	-	X
		Fx_MultiAccess	This application provides an example of Azure® RTOS FileX stack usage on STM32N6570-DK board, it demonstrates the FileX's concurrent file access capabilities. The application is designed to execute file operations on the SD card device, the code provides all required software code for handling SD card I/O operations.	-	X	-
		Fx_uSD_File_Edit	This application provides an example of Azure® RTOS FileXstack usage on STM32N6570-DK, it shows how to develop a basic SD card file operations application. The application is designed to handle SD card insertion/removal events, and depending on that state, it starts and stops file operations from and into the SD card.	-	X	-
	NetXDuo	Nx_CreditBasedShaper	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_FramePreemption_Listener	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_FramePreemption_Talker	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_Iperf	This application provides an example of Azure® RTOS NetX Duo stack usage.	-	-	X
		Nx_MQTT_Client	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	X	-
		Nx_SNTP_Client	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_TCP_Echo_Client	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_TCP_Echo_Server	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Applications	NetXDuo	Nx_TimeAwareShaper_Talker	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_UDP_Echo_Client	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_UDP_Echo_Server	This application provides an example of Azure® RTOS NetX/NetX Duo stack usage.	-	-	X
		Nx_WebServer	This application provides an example of Azure® RTOS NetX Duo stack usage on STM32N6570-DK board, it shows how to develop Web HTTP server based application.	-	X	-
	ROT	OEMuROT_Appli	This project provides a OEMuRoT boot path application example. Boot is performed through OEMuRoT boot path after authenticity and the integrity checks of the project firmware and project data images.	-	X	X
		OEMuROT_Boot	This project provides an OEMuRoT example. OEMuRoT boot path performs authenticity and the integrity checks of the project firmware and data images.	-	X	X
	ThreadX	Tx_LowPower	This application provides an example of Azure® RTOS ThreadX stack usage, it shows how to develop an application using ThreadX LowPower feature.	-	-	X
		Tx_SecureLEDToggle_TrustZone	This application provides an example of Azure® RTOS ThreadX stack usage, it shows how to develop an application using the ThreadX when the TrustZone® feature is enabled.	-	-	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.	-	-	X
		Tx_Thread_Sync	This application provides an example of Azure® RTOS ThreadX stack usage. It shows how to develop an application using the ThreadX thread management APIs.	-	-	X
	USBPD	USBPD_DRP_Ux_DRD_HID_MSC	This application is a USBPD type C DRP (supporting data role swap) using Azure® RTOS.	-	X	-
		USBPD_SNK	This application is a USBPD type C provider using Azure® RTOS.	-	X	X
		USBPD_SNK_Ux_Device_HID_CDC_ACM	This application provides an example of Azure® RTOS USBX stack usage on STM32N6570-DK board, it shows how to develop USB device communication class "HID" and "CDC_ACM" based application.	-	X	-

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6570-DK	NUCLEO-N657X0-Q
Applications	USBPD	USBPD_SRC	This application is a USBPD type C provider using Azure® RTOS.	-	X	X
		USBPD_SRC_UX_Host_MSC	This application is a USBPD type C provider using Azure® RTOS.	-	X	-
	USBX	Ux_Device_Audio_2.0	This application provides an example of Azure® RTOS USBX stack usage on STM32N6570-DK board, it shows how to develop USB device communication class "AUDIO" based application.	-	X	-
		Ux_Device_Audio_2.0_Standalone	This application provides an example of standalone USBX stack usage on STM32N6570-DK board, it shows how to develop USB device "AUDIO" class based application.	-	X	-
		Ux_Device_CDC_ACM	This application provides an example of Azure® RTOS USBX stack usage on NUCLEO-N657X0-Q board, it shows how to develop USB device communication class "CDC_ACM" based application.	-	-	X
		Ux_Device_CDC_ECM	This application provides an example of Azure® RTOS CDC_ECM stack usage on STM32N6570-DK board, it shows how to run web HTTP server based application stack over USB interface.	-	X	-
		Ux_Device_HID	This application provides an example of Azure® RTOS USBX stack usage on NUCLEO-N657X0-Q board, it shows how to develop USB device human interface "HID" mouse based application.	-	-	X
		Ux_Device_HID_CDC_ACM	This application provides an example of Azure® RTOS USBX stack usage on NUCLEO-N657X0-Q board, it shows how to develop USB device communication class "HID" and "CDC_ACM" based application.	-	-	X
		Ux_Device_MSC	This application provides an example of Azure® RTOS USBX stack usage on STM32N6570-DK board, it shows how to develop USB device mass storage class based application.	-	X	-
		Ux_Device_Video	This application provides an example of Azure® RTOS USBX stack usage on STM32N6570-DK board, it shows how to develop USB device video based application.	-	X	-
		Ux_Host_Audio_2.0	This application provides an example of Azure® RTOS USBX stack usage. It shows how to develop USB host audio able to enumerate and communicate with a device audio speaker 1.0/2.0.	-	X	-
		Ux_Host_CDC_ACM	This application provides an example of Azure® RTOS USBX stack usage.	-	X	-

Level	Module name	Project name	Description	STM32N6_CUSTOM_HW	STM32N6 570-DK	NUCLEO-N657X0-Q
Applications	USBX	Ux_Host_DualClass	This application provides an example of Azure® RTOS USBX stack usage.	-	X	-
		Ux_Host_HID	This application provides an example of Azure® RTOS USBX stack usage.	-	-	X
		Ux_Host_MSC	This application provides an example of Azure® RTOS USBX stack usage. It shows how to develop USB host mass Storage "MSC" able to enumerate and communicate with a removable USB flash disk.	-	X	-
		Ux_Host_VIDEO	This application provides an example of Azure RTOS USBX stack usage. It shows how to develop USB Host Video able to enumerate and communicate with a webcam.	-	X	-
	VENC	VENC_JPEG_Encoding	This project demonstrates the use of the STM32N6 video encoder peripheral for JPEG encoding.It is targeted to run on STM32N657xx device on STM32N6570-DK board from STMicroelectronics.	-	X	-
		VENC_RTSP_Server	This application provides an example of the H264 video encoder streamed through Azure® RTOS NetX/NetX Duo on STM32N6570-DK board.	-	X	-
		VENC_SDCard	This project demonstrates the use of the STM32N6 video encoder and the camera pipeline.It is targeted to run on STM32N657xx device on STM32N6570-DK board from STMicroelectronics.	-	X	-
		VENC_SDCard_ThreadX	This project demonstrates the use of the STM32N6 video encoder and the camera pipeline.It is targeted to run on STM32N657xx device on STM32N6570-DK board from STMicroelectronics.	-	X	-
		VENC_USB	This application provides an example of VENC usage on STM32N6570-DK board, it shows how to develop a USB video device using the camera pipeline and VENC IP.	-	X	-
	Total number of applications: 53			0	29	24
Total number of projects: 218				3	64	151

Revision history

Table 3. **Document revision history**

Date	Version	Changes
15-Nov-2024	1	Initial release.
21-Feb-2025	2	Table 2. STM32CubeN6 firmware examples updated.

Contents

List of tables

List of figures