AN5802

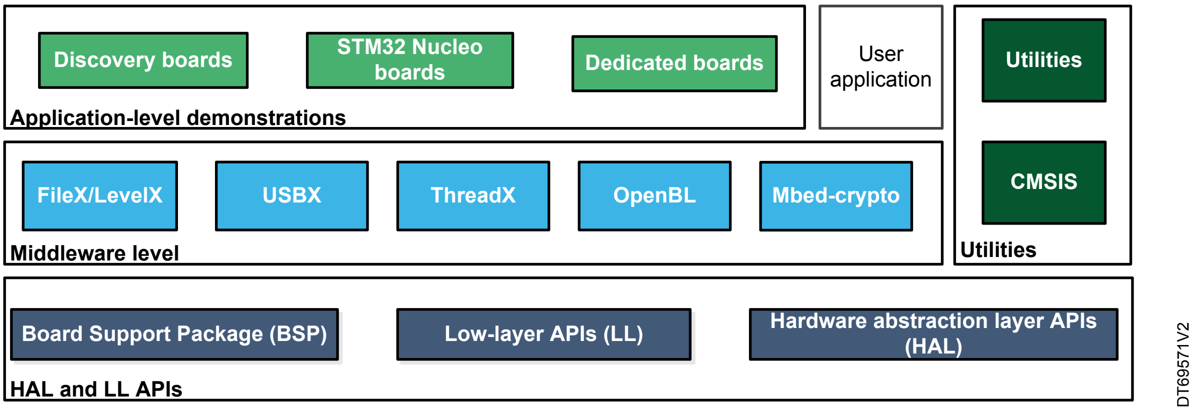
Application note

STM32Cube MCU package examples for STM32C0 Series

Introduction

The STM32CubeC0 MCU package is delivered with a rich set of examples running on STMicroelectronics boards. The examples are organized by-board, and are provided with preconfigured projects for the main-supported toolchains.

Figure 1. STM32CubeC0 firmware components



1 Reference documents

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

- Latest release of STM32CubeC0 firmware package
- Getting started with STM32CubeC0 for STM32C0 Series (UM2985)
- Description of STM32C0 HAL and low-layer drivers (UM3029)

The microcontrollers of the STM32C0 Series are based on Arm® Cortex® cores.

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

2 STM32CubeC0 examples

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

- **Examples**
  These examples only use the HAL and BSP drivers (middleware components not used). Their objective is to demonstrate the product/peripherals features and usage. They are organized per peripheral (one folder per peripheral, for example TIM). Their complexity level ranges from the basic usage of a given peripheral (such as PWM generation using timer) to the integration of several peripherals (such as how to use DAC for signal generation with synchronization from TIM6 and DMA). The usage of the board resources is reduced to the strict minimum.
- **Examples_LL**
  These examples only use the LL drivers (HAL and middleware components not used). They offer an optimum implementation of typical use cases of the peripheral features and configuration procedures. The examples are organized per peripheral (a folder for each peripheral, such as TIM). It runs exclusively on the Nucleo board.
- **Examples_MIX**
  These examples only use HAL, BSP, and LL drivers (middleware are not used). They aim to demonstrate how to use both HAL and LL APIs in the same application, to combine the advantages of both APIs (HAL offers high level and functionalities oriented APIs, with high portability level and hide product or IPs complexity to the final user. While LL offers low-level APIs at registers level with better optimization).
  The examples are organized per peripheral (a folder for each peripheral, ex. TIM). It runs exclusively on the Nucleo board.
- **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 Azure® RTOS USBX). The integration of applications that use several middleware stacks is also supported.
- **Template project**
  The template project is provided to enable the user to quickly build a firmware application using HAL and BSP drivers on a given board.
- **Template_LL project**
  The template LL projects are provided to enable the user to quickly build a firmware application using LL drivers on a given board.
- **Template_Board project**
  The template board project provides a reference template for a given board based on HAL and BSP drivers. This template was created from STM32CubeMX using the "Start My project from ST board" feature."

The examples are located under *STM32Cube_FW_C0_VX.Y.Z\Projects*.

All the examples provided have the same project structure:

- *\Inc* folder, which contains all the header files.
- *\Src* folder, which contains the sources code.
- *\EWARM*, *\MDK-ARM* and */STM32CubeIDE* folders, which contain the preconfigured project for each toolchain.
- readme.html file, which describes the example behavior and the environment required to run the example. To run the example, proceed as follows:
  – Open the example using your preferred toolchain.
  – Rebuild all files and load the image into target memory.
  – Run the example by following the readme.html instructions.

Note: Refer to "Development toolchains and compilers" and "Supported devices and evaluation, Nucleo, and discovery boards" section of the firmware package release notes to know about more about the software/ hardware environment used for the MCU package development and validation. The correct operation of the provided examples is not guaranteed in other environments, for example when using different compiler or board versions.

The examples can be customized to run on any compatible hardware: simply update the BSP drivers for the selected board, if it has the same hardware functions (LED, LCD display, push buttons … etc.). The BSP is based on a modular architecture that allows it to be ported easily to any hardware by just implementing the low-level routines.

The table below contains the list of examples provided within the STM32Cube_FW_C0 MCU package.

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

Those projects can be opened with this tool to modify the projects itself. The others projects are manually created to demonstrate the product features.

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

Table 1. STM32CubeC0 firmware examples

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Templates_Board	-	Starter project	This project provides a reference template for the NUCLEO-C071RB board based on the STM32Cube HAL API and the BSP drivers that can be used to build any firmware application.	-	-	MX	MX	MX	-
	Total number of templates_board: 3			0	0	1	1	1	0
Templates	-	Starter project	This projects provides a reference template that can be used to build any firmware application.	MX	MX	MX	MX	MX	MX
	Total number of templates: 6			1	1	1	1	1	1
ROT_Provisioning	-	OEMiSB	This section provides an overview of the available scripts for OEMiSB boot path.	-	-	-	X	-	-
	Total number of rot_provisioning: 1			0	0	0	1	0	0
Templates_LL	-	Starter project	This projects provides a reference template through the LL API that can be used to build any firmware application.	MX	MX	MX	MX	MX	MX
	Total number of templates_ll: 6			1	1	1	1	1	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 conversion data is outside the window thresholds.	-	-	-	-	MX	-
		ADC_ContinuousConversion_TriggerSW_LowPower	How to use ADC to convert a single channel with ADC low power features auto wait and auto power-off.	-	-	MX	-	-	-
		ADC_Discontinuous Conversion_TriggerSW	This example describes how to use an ADC peripheral to perform multiple conversion from different ADC channel, one at a time after each software trigger.	-	-	-	-	MX	-
		ADC_MultiChannel SingleConversion	How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence.	-	-	MX	-	MX	MX
		ADC_Oversampling	How to use an ADC peripheral with oversampling.	-	-	MX	-	-	-
		ADC_SingleConversion_TriggerSW_IT	How to use ADC to convert a single channel at each SW start, conversion performed using programming model: interrupt.	-	-	-	-	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	-

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples	ADC	ADC_SingleConversion_Trigger_EXTI	How to use ADC to convert a single channel at each trigger EXTI, conversion performed using programming model: interrupt.	-	-	-	-	MX	-
	CORTEX	CORTEXM_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	-
		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	MX	-	-
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	-	-	-	-	MX	MX
	CRC	CRC_Example	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the CRC code of a given buffer of 32-bit data words, using a fixed generator polynomial (0x4C11DB7).	-	-	MX	MX	MX	-
	DMA	DMA_FLASHToRAM	This example describes how to use DMA to transfer a word data buffer from Flash memory to embedded SRAM through the HAL API.	-	-	-	-	MX	-
	FDCAN	FDCAN_Adaptive_Bitrate_Receiver	This example describes how to configure the FDCAN peripheral to adapt to different CAN bit rates using restricted mode.	-	-	MX	-	-	-
		FDCAN_Adaptive_Bitrate_Transmitter	This example describes how to configure the FDCAN peripheral to adapt to different CAN bit rates using restricted mode.	-	-	MX	-	-	-
		FDCAN_Classic_Frame_Networking	This example describes how to configure the FDCAN peripheral to send and receive Classic CAN frames between two FDCAN units.	-	-	MX	-	-	-
		FDCAN_Com_IT	This example describes how to configure the FDCAN peripheral to achieve interrupt process communication between two FDCAN units.	-	-	MX	-	-	-
		FDCAN_Com_Polling	This example describes how to configure the FDCAN peripheral to achieve polling process communication between two FDCAN units.	-	-	MX	-	-	-
		FDCAN_Loopback	This example describes how to configure the FDCAN peripheral to operate in loopback mode.	-	-	MX	-	-	-
		FDCAN_Power_Down	This example describes the functionality of the power down mode in the FDCAN peripheral.	-	-	MX	-	-	-

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples	FLASH	FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal flash memory.	MX	MX	MX	MX	MX	MX
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	MX	MX	-	MX	MX	MX
		GPIO_EXTI_origianl	How to configure external interrupt lines.	-	-	-	MX	-	-
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	MX	MX	MX	-	MX	MX
	HAL	HAL_TimeBase	How to customize HAL using a general-purpose timer as main source of time base, instead of Systick.	MX	MX	-	-	-	MX
		HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as main source of time base instead of Systick.	-	-	-	MX	-	-
	I2C	I2C_TwoBoards_AdvComIT	How to handle several I2C data buffer transmission/reception between a master and a slave device using an interrupt.	-	-	MX	-	-	MX
		I2C_TwoBoards_ComDMA	How to handle I2C data buffer transmission/ reception between two boards via DMA.	-	-	MX	-	MX	MX
		I2C_TwoBoards_ComIT	How to handle I2C data buffer transmission/ reception between two boards, using an interrupt.	-	-	MX	-	MX	-
		I2C_WakeUpFromStop	How to handle I2C data buffer transmission/ reception between two boards, using an interrupt when the device is in Stop mode.	-	-	-	-	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	-	-	-
	PWR	PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	MX	MX	-	MX	MX	MX
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using an external reset or the WKUP pin.	MX	MX	MX	MX	MX	MX
		PWR_STOP	This example shows how to enter Stop mode and wake up from this mode using an interrupt.	-	-	-	MX	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	MX	-

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples	RCC	RCC_LSEConfig	Enabling/disabling of the low-speed external(LSE) RC oscillator (about 32 KHz) at run time, using the RCC HAL API.	MX	-	-	-	-	MX
		RCC_LSIConfig	How to enable/disable the low-speed internal (LSI) RC oscillator (about 32 KHz) at run time, using the RCC HAL API.	-	MX	-	-	-	-
		RCC_SwitchClock	Switch of the system clock (SYSCLK) from Low frequency clock to high frequency clock, using the RCC HAL API.	-	MX	-	-	-	-
	RTC	RTC_Alarm	Configuration and generation of an RTC alarm using the RTC HAL API.	-	MX	-	-	-	-
		RTC_Calendar	Configuration of the calendar using the RTC HAL API.	MX	MX	-	-	-	MX
		RTC_TimeStamp	Configuration of the RTC HAL API to demonstrate the timestamp feature.	-	MX	MX	MX	-	MX
	SMBUS	SMBUS_TwoBoards_ComIT_Master	How to handle SMBUS data buffer transmission/reception between two boards, in interrupt mode.	-	-	-	-	MX	-
		SMBUS_TwoBoards_ComIT_Slave	How to handle SMBUS data buffer transmission/reception between two boards, in interrupt mode.	-	-	-	-	MX	-
	SPI	SPI_FullDuplex_ComDMA_Master	Data buffer transmission/reception between two boards via SPI using DMA.	-	-	-	-	-	MX
		SPI_FullDuplex_ComDMA_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	-	-	-	-	-	MX
		SPI_FullDuplex_CrcComIT_Master	Data buffer transmission/reception with CRC between two boards via SPI using IT.	-	-	MX	-	MX	-
		SPI_FullDuplex_CrcComIT_Slave	Data buffer transmission/reception with CRC between two boards via SPI using IT.	-	-	MX	-	MX	-
		SPI_FullDuplex_Loopback_ComDMA	Data buffer transmission/reception with a single board via SPI using DMA.	-	-	MX	-	-	-
	TIM	TIM_Encoder	This example shows how to configure the TIM1 peripheral in encoder mode to determinate the rotation direction.	-	-	-	-	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 5 presses.	-	-	-	-	MX	-

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples	TIM	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	-
	UART	UART_AutoBaudrate_Detection	How to use the HAL UART API for detecting automatically the baud rate.	-	-	MX	-	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	MX	-	MX
		UART_ReceptionToIdle_CircularDMA	How to use the HAL UART API for reception to IDLE event in circular DMA mode.	-	-	-	-	MX	-
		UART_TwoBoards_ComIT	UART transmission (transmit/receive) in Interrupt mode between two boards.	-	-	-	-	MX	-
		UART_WakeUpFromStop	This example shows how to configure an UART to wake up the MCU from STOP1 mode when a given stimulus is received.	-	-	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	-	-	-
	Total number of examples: 109			8	8	32	12	33	16
Examples_LL	ADC	ADC_AnalogWatchdog_Init	How to use an ADC peripheral with an ADC analog watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds.	-	-	MX	MX	-	MX
		ADC_ContinuousConversion_TriggerSW_Init	How to use an ADC peripheral to convert a single channel continuously, from a software start.	-	-	-	MX	-	MX
		ADC_DiscontinuousConversion_TriggerSW	How to use an ADC peripheral to perform multiple conversion from different ADC channel, one at a time after each software trigger.	-	-	-	-	MX	-
		ADC_MultiChannelSingleConversion_Init	How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence.	-	-	MX	MX	-	-

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples_LL	ADC	ADC_Oversampling_Init	How to use an ADC peripheral with oversampling.	-	-	-	MX	-	MX
		ADC_SingleConversion_TriggerSW_DMA_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel at each software start. Converted data is transferred by DMA into a table in RAM memory.	-	-	-	MX	-	MX
		ADC_SingleConversion_TriggerSW_IT_Init	How to use ADC to convert a single channel at each SW start, conversion performed using programming model: interrupt.	-	-	MX	MX	-	MX
		ADC_SingleConversion_TriggerSW_Init	How to use ADC to convert a single channel at each SW start, conversion performed using programming model: polling.	-	-	-	MX	-	MX
		ADC_SingleConversion_TriggerTimer_DMA_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel at each trigger event from a timer. Converted data is transferred by DMA into a table in RAM memory.	-	-	-	MX	MX	MX
	CORTEX	CORTEX_MPU	Presentation of the MPU feature. This example configures a memory area as privileged read-only, and attempts to perform read and write operations in different modes.	-	-	-	-	-	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	MX	MX	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	-	MX
	DMA	DMA_CopyFromFlashToMemory_Init	How to use a DMA channel to transfer a word data buffer from Flash memory to embedded SRAM. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	MX	MX	-	MX	-	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 Joystick Selection push-button is pressed. This example is based on the STM32C0xx LL API. Peripheral initialization is done using LL initialization function to demonstrate LL init usage.	MX	MX	-	MX	-	MX
	GPIO	GPIO_InfiniteLedToggling_Init	How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms.	MX	MX	-	MX	-	MX

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples_LL	I2C	I2C_OneBoard_AdvCommunication_DMAAndIT_Init	How to exchange data between an I2C master device in DMA mode and an I2C slave device in interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	-	-	MX	-	-
		I2C_OneBoard_Communication_DMAAndIT_Init	How to transmit data bytes from an I2C master device using DMA mode to an I2C slave device using interrupt mode. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	-	-	MX	-	-
		I2C_OneBoard_Communication_IT_Init	How to handle the reception of one data byte from an I2C slave device by an I2C master device. Both devices operate in interrupt mode. The peripheral is initialized with LL 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	MX	-
		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	MX	-	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	MX	-	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	-	-	-
		I2C_TwoBoards_WakeUpFromStop_IT_Init	How to handle the reception of a data byte from an I2C slave device in Stop0 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	MX	-	MX

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples_LL	I2S	I2S_Transmit_ComDMA_Master_Init	How to handle an I2S transmission using DMA protocol in Master mode.	-	-	-	MX	-	-
	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	-	MX
	PWR	PWR_EnterStandbyMode	How to enter the Standby mode and wake up from this mode by using an external reset or a wakeup pin.	MX	MX	-	MX	-	MX
		PWR_EnterStopMode	How to enter the Stop 0 mode.	MX	MX	-	MX	-	MX
	RCC	RCC_OutputSystemClockOnMCO	Configuration of MCO pin (PA8) to output the system clock.	MX	MX	-	MX	-	MX
		RCC_UseHSEasSystemClock	Use of the RCC LL API to start the HSE and use it as system clock.	-	-	-	-	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	MX	-	-	-	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	-	-	-	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 STM32C0xx SPI LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	-	-	MX	-	-
		SPI_OneBoard_HalfDuplex_IT_Init	Configuration of 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 STM32C0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	-	MX	-	-
		SPI_TwoBoards_FullDuplex_DMA_Master_Init	Data buffer transmission and receptionvia SPI using DMA mode. This example is based on the STM32C0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-		MX	-	-	-

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples_LL	SPI	SPI_TwoBoards_FullDuplex_DMA_Slave_Init	Data buffer transmission and receptionvia SPI using DMA mode. This example is based on the STM32C0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).		-	MX	-	-	-
		SPI_TwoBoards_FullDuplex_IT_Master_Init	Data buffer transmission and receptionvia SPI using Interrupt mode. This example is based on the STM32C0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	-	MX	-	MX
		SPI_TwoBoards_FullDuplex_IT_Slave_Init	Data buffer transmission and receptionvia SPI using Interrupt mode. This example is based on the STM32C0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	-	MX	-	MX
	TIM	TIM_Central_PWM	Use of a timer peripheral to generate 3 center aligned PWMs output signals.	-	-	MX	-	-	-
		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 STM32C0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX	MX	MX	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 STM32C0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX	MX	MX	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 STM32C0xx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL Init.	-	-	MX	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 STM32C0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX	MX	-	MX

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples_LL	TIM	TIM_PWMOutput_Init	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32C0xx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL Init.	-	-	MX	MX	-	MX
		TIM_Prescaler_Update	Use of a timer peripheral to change the prescaler value on the fly.	-	-	MX	-	-	-
		TIM_TimeBase_Init	Configuration of the TIM peripheral to generate a timebase.	-	-	MX	MX	MX	MX
	USART	USART_Communication_Rx_IT_Continuous_Init	This example shows how to configure GPIO and USART peripheral for continuously receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	MX	MX	-	MX
		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	-	-	MX
		USART_Communication_Rx_IT_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	MX	MX	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	-	MX	MX
		USART_Communication_TxRx_DMA_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to/from an HyperTerminal (PC) in DMA mode. This example is based on STM32C0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	-	MX	-	MX

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples_LL	USART	USART_Communication_Tx_IT_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on STM32C0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	MX	MX	MX	MX
		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 STM32C0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	MX	MX	MX	MX
		USART_Communication_Tx_Init	This example shows how to configure GPIO and USART peripherals to send characters asynchronously to an HyperTerminal (PC) in Polling mode. If the transfer 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 STM32C0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	MX	MX	MX	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 STM32C0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	-	MX	-	MX	MX
		USART_HardwareFlowControl_Init	Configuration of GPIO and peripheral to receive characters asynchronously from an HyperTerminal (PC) in Interrupt mode with the Hardware Flow Control feature enabled. This example is based on STM32C0xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	-	MX	-	MX

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Examples_LL	USART	USART_SyncCommunication_FullDuplex_DMA_Init	Configuration of GPIO, USART, DMA and SPI peripherals to transmit bytes between a USART and an SPI (in slave mode) in DMA mode. This example is based on the STM32C0xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	-	MX	-	-
		USART_SyncCommunication_FullDuplex_IT_Init	Configuration of GPIO, USART, DMA and SPI peripherals to transmit bytes between a USART and an SPI (in slave mode) in Interrupt mode. This example is based on the STM32C0xx USART LL API (the SPI uses the DMA to receive/transmit characters sent from/received by the USART). The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	-	MX	-	MX
		USART_WakeUpFromStop_Init	Configuration of GPIO and USART1 peripherals to allow the characters received on USART_RX pin to wake up the MCU from low-power mode.	-	-	-	MX	MX	MX
	UTILS	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	MX	-	MX
	Total number of examples_ll: 148			7	8	28	47	15	43
Examples_MIX	ADC	ADC_SingleConversion_TriggerSW_IT	How to use ADC to convert a single channel at each SW start, conversion performed using programming model: interrupt.	-	-	-	-	-	MX
	CRC	CRC_PolynomialUpdate	How to use the CRC peripheral through the STM32C0xx CRC HAL and LL API.	-	-	MX	MX	MX	MX
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the STM32C0xx DMA HAL and LL API. The LL API is used for performance improvement.	-	-	-	-	-	MX
	SPI	SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	-	-	-	-	MX
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	-	-	-	-	MX
	TIM	TIM_PWMInput	Use of the TIM peripheral to measure an external signal frequency and duty cycle.	-	-	-	-	-	MX

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
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 STM32C0xx UART HAL and LL API, the LL API being used for performance improvement.	-	-	MX	-	-	MX
		UART_HyperTerminal_TxPolling_RxIT	Use of a UART to transmit data (transmit/ receive) between a board and an HyperTerminal PC application both in Polling and Interrupt modes. This example describes how to use the USART peripheral through the STM32C0xx UART HAL and LL API, the LL API being used for performance improvement.	-	-	MX	-	-	MX
	Total number of examples_mix: 13			0	0	3	1	1	8
Applications	-	OpenBootloader	This application exploits OpenBootloader Middleware to demonstrate how to develop an IAP application and how use it.	-	-	-	-	-	X
	FileX	Fx_File_Edit_Standalone	This application provides an example of FileX stack usage on NUCLEO-C071RB board, running in standalone mode (without ThreadX). It demonstrates how to create a Fat File system on the internal SRAM memory using FileX.	-	-	-	MX	-	-
		Fx_SRAM_File_Edit_Standalone	This application provides an example of FileX stack usage on NUCLEO-C031C6 board, running in standalone mode (without ThreadX). It demonstrates how to create a Fat File system on the internal SRAM memory using FileX.	-	-	MX	-	-	MX
	ROT	OEMiSB_Appli	This project provides a OEMiSB boot path application example. Boot is performed through OEMiSB boot path after integrity checks of the project firmware image.	-	-	X	X	X	-
		OEMiSB_Boot	This project provides an OEMiSB example. OEMiSB boot path performs authenticity checks of the project firmware image.	-	-	X	X	X	-
	ThreadX	Tx_CMSIS_Wrapper	This application provides an example of CMSIS RTOS adaptation layer for Azure RTOS ThreadX, it shows how to develop an application using the CMSIS RTOS 2 APIs.	X	-	-	-	-	-
		Tx_FreeRTOS_Wrapper	This application provides an example of Azure RTOS ThreadX stack usage, it shows how to develop an application using the FreeRTOS adaptation layer for ThreadX.	-	X	-	-	-	-
		Tx_LowPower	This application provides an example of Azure RTOS ThreadX stack usage, it shows how to develop an application using ThreadX LowPower feature.	-	-	MX	MX	-	MX

Level	Module Name	Project Name	Description	STM32C0316-DK	STM32C0116-DK	NUCLEO-C092RC	NUCLEO-C071RB	NUCLEO-C051C8	NUCLEO-C031C6
Applica-tions	ThreadX	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	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	-	MX	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	MX	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	-	-	MX	MX	-
	USBX	Ux_Device_CDC_ACM	This application provides an example of Azure RTOS USBX stack usage on NUCLEO-C071RB board, it shows how to develop USB Device communication Class "CDC_ACM" based application.	-	-	-	MX	-	-
		Ux_Device_HID	This application provides an example of Azure RTOS USBX stack usage on NUCLEO-C071RB board, it shows how to develop USB Device Human Interface "HID" mouse based application.	-	-	-	MX	-	-
		Ux_Device_HID_CDC_ACM	This application provides an example of Azure RTOS USBX stack usage on NUCLEO-C071RB board, it shows how to develop a composite USB device communication class "HID" and "CDC_ACM" based application.	-	-	-	MX	-	-
		Ux_Device_HID_Standalone	This application provides an example of Azure RTOS USBX stack usage on NUCLEO-C071RB board, it shows how to develop USB Device Human Interface "HID" mouse based bare metal application.	-	-	-	MX	-	-
		Ux_Host_CDC_ACM	This application provides an example of Azure RTOS USBX stack usage.	-	-	-	MX	-	-
		Ux_Host_HID_Standalone	This application provides an example of Azure RTOS USBX stack usage.	-	-	-	MX	-	-
	Total number of applications: 32			3	1	6	14	4	4
Total number of projects: 318				20	19	16	76	56	73

Revision history

Table 2. **Document revision history**

Date	Version	Changes
08-Jul-2022	1	Initial release.
19-Aug-2024	2	Updated: • Section 2: STM32CubeC0 examples • Figure 1. STM32CubeC0 firmware components • Table 1. STM32CubeC0 firmware examples
10-Dec-2024	3	Updated: • Table 1. STM32CubeC0 firmware examples • Figure 1. STM32CubeC0 firmware components
07-Mar-2025	4	Updated: • Table 1. STM32CubeC0 firmware examples

Contents