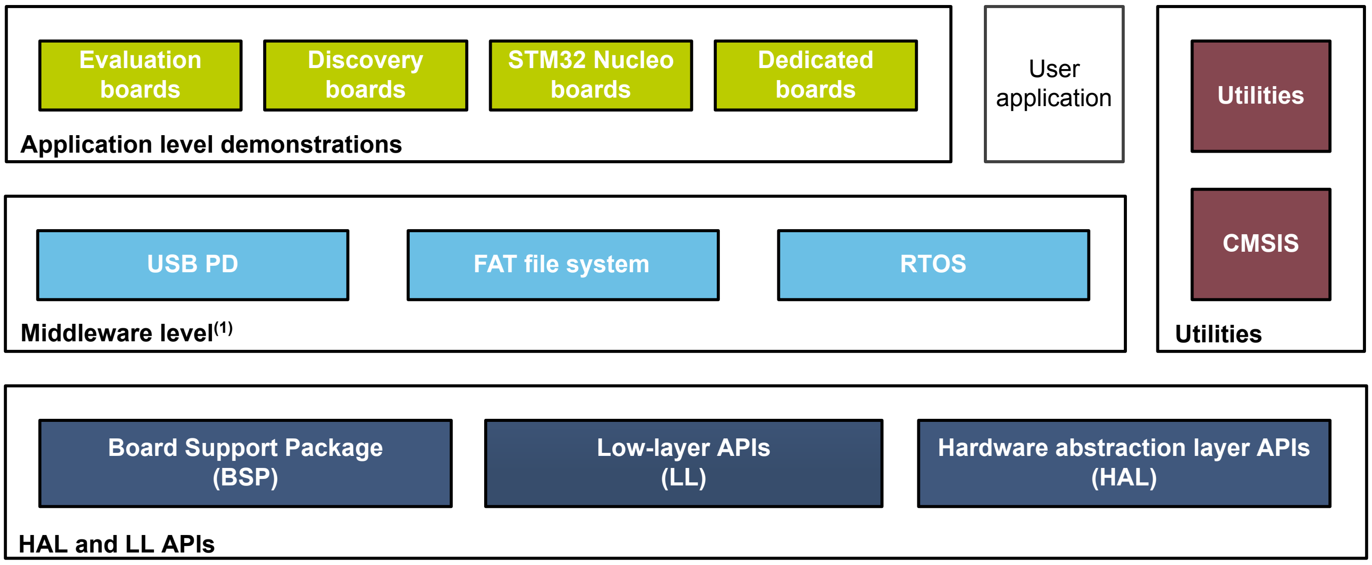
STM32Cube firmware examples for STM32G0 Series

Introduction

The STM32CubeG0 MCU Package is delivered with a rich set of examples running on STMicroelectronics boards. The examples are organized by board and provided with preconfigured projects for the main supported toolchains (refer to Figure 1).

In the STM32CubeG0 MCU Package, most of examples and applications projects are generated with the STM32CubeMX tool (starting from version v5.0.0) to initialize the system, peripherals, and middleware stacks. The user can open the provided *ioc* file in STM32CubeMX to modify the settings, and add additional peripherals, middleware components or both, to build his final application. For more information about STM32CubeMX, refer to the *STM32CubeMX for STM32 configuration and initialization C code generation* user manual (UM1718).

Figure 1. STM32CubeG0 firmware components



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

1 Reference documents

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

- Latest release of the STM32CubeG0 MCU Package for the 32-bit microcontrollers in the STM32G0 Series based on the Arm® Cortex®-M processor
- *Getting started with STM32CubeG0 for STM32G0 Series* (UM2303)
- *STM32CubeG0 Nucleo demonstration firmware* (UM2308)
- *STM32CubeG0 STM32G081B-EVAL demonstration firmware* (UM2321)
- *STM32G071B-DISCO USB-C Discovery kit* (UM2546)
- *Description of STM32G0 HAL and low-layer drivers* (UM2319)
- *STM32Cube USBPD stack user manual* (UM2552)
- *Developing applications on STM32Cube with FatFS* (UM1721)
- *Developing applications on STM32Cube with RTOS* (UM1722)

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

2 STM32CubeG0 examples

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

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

  The examples are organized per peripheral (one folder for each peripheral, e.g. TIM) and run exclusively on Nucleo board.
- **Applications**
  The applications demonstrate the product performance and how to use the available middleware stacks. They are organized either by middleware (a folder per middleware, e.g. USB Host) or by product feature that require high-level firmware bricks (e.g. Audio). The integration of applications that use several middleware stacks is also supported.
- **Demonstrations**
  The demonstrations aim at integrating and running the maximum number of peripherals and middleware stacks to showcase the product features and performance.
- **Template project**
  The template project is provided to allow the user to quickly build a firmware application using HAL and BSP drivers on a given board.
- **Template_LL project**
  The template LL projects are provided to allow the user to quickly build a firmware application using LL drivers on a given board.

The examples are located under *STM32Cube_FW_G0_VX.Y.Z\Projects\*. They all have the same structure:

- *\Inc* folder, containing all header files
- *\Src* folder, containing the sources code
- *\EWARM, \MDK-ARM and \SW4STM32* folders, containing the preconfigured project for each toolchain
- *readme.txt* file, describing the example behavior and the environment required to run the example
- **.ioc* file that allows users to open most of firmware examples within STM32CubeMX (starting from STM32CubeMX version v5.0.0)

To run the example, proceed as follows:

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

Note: Refer to "Development toolchains and compilers" and "Supported devices and evaluation boards" sections of the firmware package release notes to know more about the software/hardware environment used for the MCU

Package development and validation. The correct operation of the provided examples is not guaranteed in other environments, for example when using different compiler or board versions.

The examples can be tailored to run on any compatible hardware: simply update the BSP drivers for your board, provided it has the same hardware functions (LED, LCD display, pushbuttons, etc.). 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 STM32CubeG0 MCU Package.

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

Table 1. STM32CubeG0 firmware examples

Level	Module Name	Project Name	Description	NUCLEO-G070RB[1]	NUCLEO-G071RB[1]	STM32G081B-EVAL[1]	STM32G071B-DISCO[1]
Templates	-	Starter project	This projects provides a reference template that can be used to build any firmware application.	MX	MX	MX	MX
	Total number of templates: 4			**1**	**1**	**1**	**1**
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
	Total number of templates_ll: 4			**1**	**1**	**1**	**1**
Examples	ADC	ADC_AnalogWatchdog	How to use the ADC peripheral to perform conversions with an analog watchdog and out-of-window interrupts enabled.	-	MX	-	-
		ADC_MultiChannelSingleConversion	Use ADC to convert a several channels using sequencer in discontinuous mode, conversion data are transferred by DMA into an array, indefinitely (circular mode).	MX	MX	MX	-
		ADC_Oversampling	Use ADC to convert a single channel but using oversampling feature to increase resolution.	-	-	MX	-
		ADC_SingleConversion_TriggerSW_IT	Use ADC to convert a single channel at each SW start, conversion performed using programming model: interrupt Example configuration: ADC is configured to convert a single channel, in single conversion mode, from SW trigger.	-	MX	-	-
		ADC_SingleConversion_TriggerTimer_DMA	Use ADC to convert a single channel at each trig from timer, conversion data are transferred by DMA into an array, indefinitely (circular mode).	-	MX	-	-
	BSP	BSP_Example	This example provides a description of how to use the different BSP drivers.	-	-	MX	-
	CEC	CEC_DataExchange_Device_1	This example shows how to configure and use the CEC peripheral to receive and transmit messages.	-	-	MX	-
		CEC_DataExchange_Device_2	This example shows how to configure and use the CEC peripheral to receive and transmit messages.	-	-	MX	-
		CEC_ListenMode_Device_1	This example shows how to configure and use the CEC peripheral to receive and transmit messages between two boards while a third one (the spy device) listens but doesn't acknowledge the received messages.	-	-	MX	-
		CEC_ListenMode_Device_2	This example shows how to configure and use the CEC peripheral to receive and transmit messages between two boards while a third one (the spy device) listens but doesn't acknowledge the received messages.	-	-	MX	-
		CEC_ListenMode_Device_3	This example shows how to configure and use the CEC peripheral to receive and transmit messages between two boards while a third one (the spy device) listens but doesn't acknowledge the received messages.	-	-	MX	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB(1)	NUCLEO-G071RB(1)	STM32G081B-EVAL(1)	STM32G071B-DISCO(1)
Examples	CEC	CEC_MultiAddress_Device_1	This example shows how to configure and use the CEC peripheral to receive and transmit messages in the case where one device supports two distinct logical addresses at the same time.	-	-	MX	-
		CEC_MultiAddress_Device_2	This example shows how to configure and use the CEC peripheral to receive and transmit messages in the case where one device supports two distinct logical addresses at the same time.	-	-	MX	-
	COMP	COMP_CompareGpioVsVrefInt_IT	How to configure the COMP peripheral to compare the external voltage applied on a specific pin with the Internal Voltage Reference.	-	MX	MX	-
		COMP_CompareGpioVsVrefInt_Window_IT	How to make window comparator using the COMP peripherals in window mode.	-	MX	-	-
	CORTEX	CORTEXM_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	MX	-	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	MX	-	MX
		CORTEXM_ProcessStack	How to modify the Thread mode stack. Thread mode is entered on reset, and can be entered as a result of an exception return.	MX	MX	-	MX
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	MX	MX	-	MX
	CRC	CRC_Bytes_Stream_7bit_CRC	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes 7-bit CRC codes derived from buffers of 8-bit data (bytes). The user-defined generating polynomial is manually set to 0x65, that is, X^7 + X^6 + X^5 + X^2 + 1, as used in the Train Communication Network, IEC 60870-5[17].	MX	MX	MX	MX
		CRC_Data_Reversing_16bit_CRC	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes a 16-bit CRC code derived from a buffer of 8-bit data (bytes). Input and output data reversal features are enabled. The user-defined generating polynomial is manually set to 0x1021, that is, X^16 + X^12 + X^5 + 1 which is the CRC-CCITT generating polynomial.	MX	MX	MX	MX
		CRC_Example	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the CRC code of a given buffer of 32-bit data words, using a fixed generator polynomial (0x4C11DB7).	MX	MX	MX	MX
		CRC_UserDefinedPolynomial	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the 8-bit CRC code for a given buffer of 32-bit data words, based on a user-defined generating polynomial.	MX	MX	MX	MX
	CRYP	CRYP_AESModes	How to use the CRYP peripheral to encrypt and decrypt data using AES in chaining modes (ECB, CBC, CTR).	-	-	MX	-
		CRYP_DMA	How to use the CRYP peripheral to encrypt and decrypt data using the AES-128 algorithm with ECB chaining mode in DMA mode.	-	-	MX	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[(1)]	NUCLEO-G071RB[(1)]	STM32G081B-EVAL[(1)]	STM32G071B-DISCO[(1)]
Examples	DAC	DAC_SignalsGeneration	How to use the DAC peripheral to generate several signals using the DMA controller and the DAC internal wave generator.	-	MX	MX	-
		DAC_SimpleConversion	How to use the DAC peripheral to do a simple conversion.	-	MX	MX	-
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the HAL API.	MX	MX	MX	MX
	FLASH	FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal Flash memory.	MX	MX	MX	MX
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	MX	MX	-	-
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	MX	MX	MX	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_RTC_ALARM	How to customize HAL using RTC alarm as main source of time base, instead of Systick.	MX	MX	MX	-
		HAL_TimeBase_RTC_WKUP	How to customize HAL using RTC wakeup 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	MX	MX	-
	I2C	I2C_TwoBoards_AdvComIT	How to handle I2C data buffer transmission/reception between two boards, 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	MX	-
		I2C_TwoBoards_ComPolling	How to handle I2C data buffer transmission/reception between two boards, in polling mode.	MX	MX	MX	-
		I2C_TwoBoards_RestartAdvComIT	How to perform multiple I2C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	MX	MX	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[(1)]	NUCLEO-G071RB[(1)]	STM32G081B-EVAL[(1)]	STM32G071B-DISCO[(1)]
Examples	I2C	I2C_TwoBoards_RestartComIT	How to handle single I2C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	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	MX	-	-
	IWDG	IWDG_Reset	How to handle the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset laps of time.	MX	MX	MX	MX
		IWDG_WindowMode	How to periodically update the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset laps of time.	MX	MX	MX	MX
	LPTIM	LPTIM_PWMExternalClock	How to configure and use, through the HAL LPTIM API, the LPTIM peripheral using an external counter clock, to generate a PWM signal at the lowest power consumption.	-	MX	MX	-
		LPTIM_PWM_LSE	How to configure and use, through the HAL LPTIM API, the LPTIM peripheral using LSE as counter clock, to generate a PWM signal, in a low-power mode.	-	-	MX	-
		LPTIM_PulseCounter	How to configure and use, through the LPTIM HAL API, the LPTIM peripheral to count pulses.	-	MX	MX	-
		LPTIM_Timeout	How to implement, through the HAL LPTIM API, a timeout with the LPTIMER peripheral, to wake up the system from a low-power mode.	-	MX	MX	-
	PWR	PWR_LPRUN	How to enter and exit the Low-power run mode.	MX	MX	MX	MX
		PWR_LPSLEEP	How to enter the Low-power sleep mode and wake up from this mode by using an interrupt.	MX	MX	MX	-
		PWR_PVD	How to configure the programmable voltage detector by using an external interrupt line. External DC supply must be used to supply Vdd.	-	MX	MX	-
		PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	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	-
	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	-
		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	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB(1)	NUCLEO-G071RB(1)	STM32G081B-EVAL(1)	STM32G071B-DISCO(1)
Examples	RCC	RCC_LSIConfig	Enabling/disabling of the low-speed internal (LSI) RC oscillator (about 32 KHz) at run time, using the RCC HAL API.	MX	MX	-	-
		RCC_SwitchClock	Switch of the system clock (SYSCLK) from Low frequency clock to high frequency clock, using the RCC HAL API.	-	-	MX	MX
	RNG	RNG_MultiRNG	Configuration of the RNG using the HAL API. This example uses the RNG to generate 32-bit long random numbers.	-	-	MX	-
		RNG_MultiRNG_IT	Configuration of the RNG using the HAL API. This example uses RNG interrupts to generate 32-bit long random numbers.	-	-	MX	-
	RTC	RTC_Alarm	Configuration and generation of an RTC alarm using the RTC HAL API.	MX	MX	MX	-
		RTC_Calendar	Configuration of the calendar using the RTC HAL API.	MX	MX	MX	-
		RTC_InternalTimeStamp	Demonstration the internal timestamp feature using the RTC HAL API.	-	-	MX	-
		RTC_LSI	Use of the LSI clock source autocalibration to get a precise RTC clock.	MX	MX	MX	-
		RTC_LowPower_STANDBY	How to enter STANDBY mode and wake up from this mode using the RTC alarm event.	-	-	MX	-
		RTC_Tamper	Configuration of the RTC HAL API to write/read data to/from RTC Backup registers.	MX	MX	MX	-
		RTC_TimeStamp	Configuration of the RTC HAL API to demonstrate the timestamp feature.	MX	MX	MX	-
	SMBUS	SMBUS_TSENSOR	This example shows how to ensure SMBUS Data buffer transmission and reception with IT. The communication is done with a SMBUS temperature sensor.	-	-	MX	-
	SPI	SPI_FullDuplex_ComDMA_Master	Data buffer transmission/reception between two boards via SPI using DMA.	MX	MX	MX	-
		SPI_FullDuplex_ComDMA_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	MX	MX	MX	-
		SPI_FullDuplex_ComIT_Master	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	MX	MX	MX	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[(1)]	NUCLEO-G071RB[(1)]	STM32G081B-EVAL[(1)]	STM32G071B-DISCO[(1)]
Examples	SPI	SPI_FullDuplex_ComIT_Slave	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	MX	MX	MX	-
		SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI using Polling mode.	MX	MX	MX	-
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI using Polling mode.	MX	MX	MX	-
	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	MX	MX	-
		TIM_DMABurst	How to update the TIMER channel 1 period and duty cycle using the TIMER DMA burst feature.	MX	MX	MX	-
		TIM_ExtTriggerSynchro	This example shows how to synchronize TIM peripherals in cascade mode with an external trigger.	MX	MX	MX	-
		TIM_InputCapture	How to use the TIM peripheral to measure an external signal frequency.	MX	MX	MX	-
		TIM_OCActive	Configuration of the TIM peripheral in Output Compare Active mode (when the counter matches the capture/compare register, the corresponding output pin is set to its active state).	MX	MX	MX	-
		TIM_OCInactive	Configuration of the TIM peripheral in Output Compare Inactive mode with the corresponding Interrupt requests for each channel.	MX	MX	MX	-
		TIM_OCToggle	Configuration of the TIM peripheral to generate four different signals at four different frequencies.	MX	MX	MX	-
		TIM_OnePulse	Use of the TIM peripheral to generate a single pulse when an external signal rising edge is received on the timer input pin.	X	X	X	-
		TIM_PWMInput	How to use the TIM peripheral to measure the frequency and duty cycle of an external signal.	MX	MX	MX	-
		TIM_PWMOutput	Configuration of the TIM peripheral in PWM (pulse width modulation) mode.	MX	MX	MX	-
		TIM_TimeBase	Configuration of the TIM peripheral to generate a time base of one second with the corresponding interrupt request.	MX	MX	MX	-
	UART	LPUART_WakeUpFromStop	Configuration of an LPUART to wake up the MCU from Stop mode when a given stimulus is received.	-	MX	MX	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB(1)	NUCLEO-G071RB(1)	STM32G081B-EVAL(1)	STM32G071B-DISCO(1)
Examples	UART	UART_HyperTerminal_DMA	UART transmission (transmit/receive) in DMA 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_TwoBoards_ComDMA	UART transmission (transmit/receive) in DMA mode between two boards.	MX	MX	MX	-
		UART_TwoBoards_ComIT	UART transmission (transmit/receive) in Interrupt mode between two boards.	MX	MX	MX	-
		UART_TwoBoards_ComPolling	UART transmission (transmit/receive) in Polling mode between two boards.	MX	MX	MX	-
		WWDG_Example	Configuration of the HAL API to periodically update the WWDG counter and simulate a software fault that generates an MCU WWDG reset when a predefined time period has elapsed.	MX	MX	MX	MX
	Total number of examples: 222			**59**	**71**	**76**	**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	-	-
		ADC_ContinuousConversion_TriggerSW	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	X	X	-	-
		ADC_ContinuousConversion_TriggerSW_Init	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	MX	MX	-	-
		ADC_ContinuousConversion_TriggerSW_LowPower_Init	How to use an ADC peripheral with ADC low-power features.	MX	MX	-	-
		ADC_MultiChannelSingleConversion	How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence.	-	X	-	-
		ADC_Oversampling_Init	How to use an ADC peripheral with ADC 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. This example uses the DMA programming model (for polling or interrupt programming models, refer to other examples).	MX	MX	-	-
		ADC_SingleConversion_TriggerSW_IT_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel, at each software start. This example uses the interrupt programming model (for polling or DMA programming models, please refer to other examples).	MX	MX	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB(1)	NUCLEO-G071RB(1)	STM32G081B-EVAL(1)	STM32G071B-DISCO(1)
Examples_LL	ADC	ADC_SingleConversion_TriggerSW_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel at each software start. This example uses the polling programming model (for interrupt or DMA programming models, please refer to other examples).	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 indefinitely transferred by DMA into a table (circular mode).	MX	MX	-	-
	COMP	COMP_CompareGpioVsVrefInt_IT	How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the internal voltage reference (VREFINT), in interrupt mode. This example is based on the STM32G0xx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		COMP_CompareGpioVsVrefInt_IT_Init	How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the the internal voltage reference (VREFINT), in interrupt mode. This example is based on the STM32G0xx COMP LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	MX	-	-
		COMP_CompareGpioVsVrefInt_OutputGpio_Init	How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the internal voltage reference (VREFINT). The comparator output is connected to a GPIO. This example is based on the STM32G0xx COMP LL API.	-	MX	-	-
		COMP_CompareGpioVsVrefInt_Window_IT_Init	How to use a pair of comparator peripherals to compare a voltage level applied on a GPIO pin to two thresholds: the internal voltage reference (VREFINT) and a fraction of the internal voltage reference (VREFINT/2), in interrupt mode. This example is based on the STM32G0xx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	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	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	-	-
		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	-	-
	DAC	DAC_GenerateConstantSignal_TriggerSW_Init	How to use the DAC peripheral to generate a constant voltage signal. This example is based on the STM32G0xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-
		DAC_GenerateConstantSignal_TriggerSW_LP_Init	How to use the DAC peripheral to generate a constant voltage signal with the DAC low-power feature sample-and-hold. To be effective, a capacitor must be connected to the DAC channel output and the sample-and-hold timings must be tuned depending on the capacitor value. This example is based on the STM32G0xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-
		DAC_GenerateWaveform_TriggerHW	How to use the DAC peripheral to generate a voltage waveform from a digital data stream transfered by DMA. This example is based on the STM32G0xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[(1)]	NUCLEO-G071RB[(1)]	STM32G081B-EVAL[(1)]	STM32G071B-DISCO[(1)]
Examples_LL	DMA	DMA_CopyFromFlashToMemory	How to use a DMA channel to transfer a word data buffer from Flash memory to embedded SRAM. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	X	X	-	-
		DMA_CopyFromFlashToMemory_Init	How to use a DMA channel to transfer a word data buffer from Flash memory to embedded SRAM. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	MX	MX	-	MX
	EXTI	EXTI_ToggleLedOnIT	How to configure the EXTI and use GPIOs to toggle the user LEDs available on the board when a user button is pressed. It is based on the STM32G0xx LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	X	X	-	-
		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 STM32G0xx LL API. Peripheral initialization is done using LL initialization function to demonstrate LL init usage.	MX	MX	-	-
	GPIO	GPIO_InfiniteLedToggling	How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32G0xx LL API. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	X	X	-	-
		GPIO_InfiniteLedToggling_Init	How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32G0xx LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	MX	MX	-	-
	I2C	I2C_OneBoard_Communication_IT	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.	X	X	-	-
		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_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	-	-
	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	-	-
	LPTIM	LPTIM_PulseCounter	How to use the LPTIM peripheral in counter mode to generate a PWM output signal and update its duty cycle. This example is based on the STM32G0xx LPTIM LL API. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	X	-	-
		LPTIM_PulseCounter_Init	How to use the LPTIM peripheral in counter mode to generate a PWM output signal and update its duty cycle. This example is based on the STM32G0xx LPTIM LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	-	MX	-	-
	LPUART	LPUART_WakeUpFromStop	Configuration of GPIO and LPUART peripherals to allow characters received on LPUART_RX pin to wake up the MCU from low-power mode. This example is based on the LPUART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
	PWR	PWR_EnterStandbyMode	How to enter the Standby mode and wake up from this mode by using an external reset or a wakeup interrupt.	MX	MX	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[1]	NUCLEO-G071RB[1]	STM32G081B-EVAL[1]	STM32G071B-DISCO[1]
Examples_LL	PWR	PWR_EnterStopMode	How to enter the STOP 0 mode.	MX	MX	-	-
	RCC	RCC_OutputSystemClockOnMCO	Configuration of MCO pin (PA8) to output the system clock.	MX	MX	-	-
		RCC_UseHSEasSystemClock	Use of the RCC LL API to start the HSE and use it as system clock.	MX	MX	-	-
		RCC_UseHSI_PLLasSystemClock	Modification of the PLL parameters in run time.	MX	MX	-	-
	RTC	RTC_Alarm	Configuration of the RTC LL API to configure and generate an alarm using the RTC peripheral. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	X	X	-	-
		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	-	-
		RTC_ExitStandbyWithWakeUpTimer_Init	Configuration of the RTC to wake up from Standby mode using the RTC Wakeup timer. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	MX	-	-
		RTC_Tamper_Init	Configuration of the Tamper using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	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_IT	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 STM32G0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	X	X	-	-
		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 STM32G0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	MX	-	-
		SPI_TwoBoards_FullDuplex_IT_Master_Init	Data buffer transmission and receptionvia SPI using Interrupt mode. This example is based on the STM32G0xx 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 STM32G0xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	MX	-	-
	TIM	TIM_BreakAndDeadtime	Configuration of the TIM peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined deadtime value, use the break feature, and lock the break and dead-time configuration.	-	X	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[1]	NUCLEO-G071RB[1]	STM32G081B-EVAL[1]	STM32G071B-DISCO[1]
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 STM32G0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	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 STM32G0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	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 STM32G0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	MX	-	-
		TIM_PWMOutput	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32G0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		TIM_PWMOutput_Init	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32G0xx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL Init.	MX	MX	-	-
		TIM_TimeBase	Configuration of the TIM peripheral to generate a timebase. This example is based on the STM32G0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	X	-	-	-
		TIM_TimeBase_Init	Configuration of the TIM peripheral to generate a timebase. This example is based on the STM32G0xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	MX	-	-
	USART	USART_Communication_Rx_IT	Configuration of GPIO and USART peripherals to receive characters from an HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	X	X	-	-
		USART_Communication_Rx_IT_Continuous_Init	This example shows how to configure GPIO and USART peripheral for continuously receiving characters from HyperTerminal (PC) in Asynchronous mode using Interrupt mode. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	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	-	-
		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	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[(1)]	NUCLEO-G071RB[(1)]	STM32G081B-EVAL[(1)]	STM32G071B-DISCO[(1)]
Examples_LL	USART	USART_Communication_TxRx_DMA	Configuration of GPIO and USART peripherals to send characters asynchronously to/from an HyperTerminal (PC) in DMA mode.	-	X	-	-
		USART_Communication_TxRx_DMA_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to/from an HyperTerminal (PC) in DMA mode. This example is based on STM32G0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	MX	MX	-	-
		USART_Communication_Tx_IT	Configuration of GPIO and USART peripherals to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on the STM32G0xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		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 STM32G0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	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 STM32G0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	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 STM32G0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	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 STM32G0xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	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 STM32G0xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	MX	-	-
		USART_SyncCommunication_FullDuplex_DMA_Init	Configuration of GPIO, USART, DMA and SPI peripherals to transmit bytes between a USART and an SPI (in slave mode) in DMA mode. This example is based on the STM32G0xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	MX	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 STM32G0xx 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	Configuration of GPIO and USART peripherals to allow the characters received on USART RX pin to wake up MCU from low power mode. This example is based on the STM32G0xx USART LL API.	-	X	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[1]	NUCLEO-G071RB[1]	STM32G081B-EVAL[1]	STM32G071B-DISCO[1]
Examples_LL	USART	USART_WakeUpFromStop1_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	-	-
		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	-	-
	UTILS	UTILS_ConfigureSystemClock	Use of UTILS LL API to configure the system clock using PLL with HSI as source clock.	MX	MX	-	-
		UTILS_ReadDeviceInfo	This example reads the UID, Device ID and Revision ID and saves them into a global information buffer.	MX	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	-	-
	Total number of examples_ll: 136			**60**	**75**	**0**	**1**
Examples_MIX	ADC	ADC_SingleConversion_TriggerSW_IT	How to use the ADC to perform a single ADC channel conversion at each software start. This example uses the interrupt programming model (for polling and DMA programming models, please refer to other examples). It is based on the STM32G0xx ADC HAL and LL API. The LL API is used for performance improvement.	MX	MX	-	-
	CRC	CRC_PolynomialUpdate	How to use the CRC peripheral through the STM32G0xx CRC HAL and LL API.	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 STM32G0xx DMA HAL and LL API. The LL API is used for performance improvement.	MX	MX	-	MX
	I2C	I2C_OneBoard_ComSlave7_10bits_IT	How to perform I2C data buffer transmission/reception between one master and two slaves with different address sizes (7-bit or 10-bit). This example uses the STM32G0xx I2C HAL and LL API (LL API usage for performance improvement) and an interrupt.	MX	MX	-	-
	SPI	SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI using Polling mode.	MX	MX	-	-
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI using Polling mode.	MX	MX	-	-
	TIM	TIM_PWMInput	Use of the TIM peripheral to measure an external signal frequency and duty cycle.	MX	MX	-	-
	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 STM32G0xx UART HAL and LL API, the LL API being used for performance improvement.	MX	MX	-	-

Level	Module Name	Project Name	Description	NUCLEO-G070RB[1]	NUCLEO-G071RB[1]	STM32G081B-EVAL[1]	STM32G071B-DISCO[1]
Examples_MIX	UART	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 STM32G0xx UART HAL and LL API, the LL API being used for performance improvement.	MX	MX	-	-
	Total number of examples_mix: 20			9	9	0	2
Applications	FatFs	FatFs_uSD_Standalone	How to use STM32Cube firmware with FatFs middleware component as a generic FAT file system module. This example develops an application that exploits FatFs features to configure a microSD drive.	MX	MX	MX	-
	FreeRTOS	FreeRTOS_Mail	How to use mail queues with CMSIS RTOS API.	MX	MX	MX	-
		FreeRTOS_Mutexes	How to use mutexes with CMSIS RTOS API.	MX	MX	MX	-
		FreeRTOS_Queues	How to use message queues with CMSIS RTOS API.	MX	MX	MX	-
		FreeRTOS_Semaphore	How to use semaphores with CMSIS RTOS API.	MX	MX	MX	-
		FreeRTOS_SemaphoreFromISR	How to use semaphore from ISR with CMSIS RTOS API.	MX	MX	MX	-
		FreeRTOS_Signal	How to perform thread signaling using CMSIS RTOS API.	MX	MX	MX	-
		FreeRTOS_SignalFromISR	This application shows the usage of CMSIS-OS Signal API from ISR context.	MX	MX	MX	-
		FreeRTOS_ThreadCreation	How to implement thread creation using CMSIS RTOS API.	MX	MX	MX	-
		FreeRTOS_Timers	How to use timers of CMSIS RTOS API.	MX	MX	MX	-
	USB-PD	USB-PD_Consumer_1port	How to create a simple type C Consumer.	-	-	MX	-
		USB-PD_Provider_1port	How to create a simple type C provider.	-	-	MX	-
	Total number of applications: 32			10	10	12	0

Level	Module Name	Project Name	Description	NUCLEO-G070RB[1]	NUCLEO-G071RB[1]	STM32G081B-EVAL[1]	STM32G071B-DISCO[1]
Demonstrations	-	Adafruit_LCD_1_8_SD_Joystick	This demonstration firmware is based on STM32Cube. It helps you to discover STM32 Cortex-M devices that can be plugged on a STM32 Nucleo board.	MX	MX	-	-
		DemoLegacy	The provided demonstration "Legacy" firmware based on STM32Cube helps you to discover STM32 Cortex-M devices that can be plugged on a STM32G081B-EVAL board.	-	-	X	-
		DemoLoader	The provided demonstration "Loader" firmware based on STM32Cube helps you to discover STM32 Cortex-M devices that can be plugged on a STM32G081B-EVAL board.	-	-	X	-
		DemoUCPD	This demonstration firmware is based on STM32Cube and describes how to use USB Power Delivery (USB-PD) feature based on STM32G081B-EVAL + MB1352 extension boards.	-	-	X	-
		USBPD_Analyzer	This demonstration firmware is based on STM32Cube and describes how to use USB Power Delivery (USB-PD) feature based on STM32G071B-DISCO board.	-	-	-	X
	Total number of demonstrations: 6			**1**	**1**	**3**	**1**
Total number of projects: 424				**141**	**168**	**93**	**22**

1. *STM32CubeMX-generated examples are highlighted with the* MX *STM32CubeMX icon. Other examples are marked with “x” .*

Revision history

Table 2. **Document revision history**

Date	Version	Changes
1-Dec-2017	1	Initial release.
15-Nov-2018	2	Document scope extended to the NUCLEO-G071RB board. STM32CubeMX-generated examples highlighted in *Table 1. STM32CubeG0 firmware examples*.
26-Feb-2019	3	Document scope extended to the STM32G071B-DISCO board.

Contents

List of tables