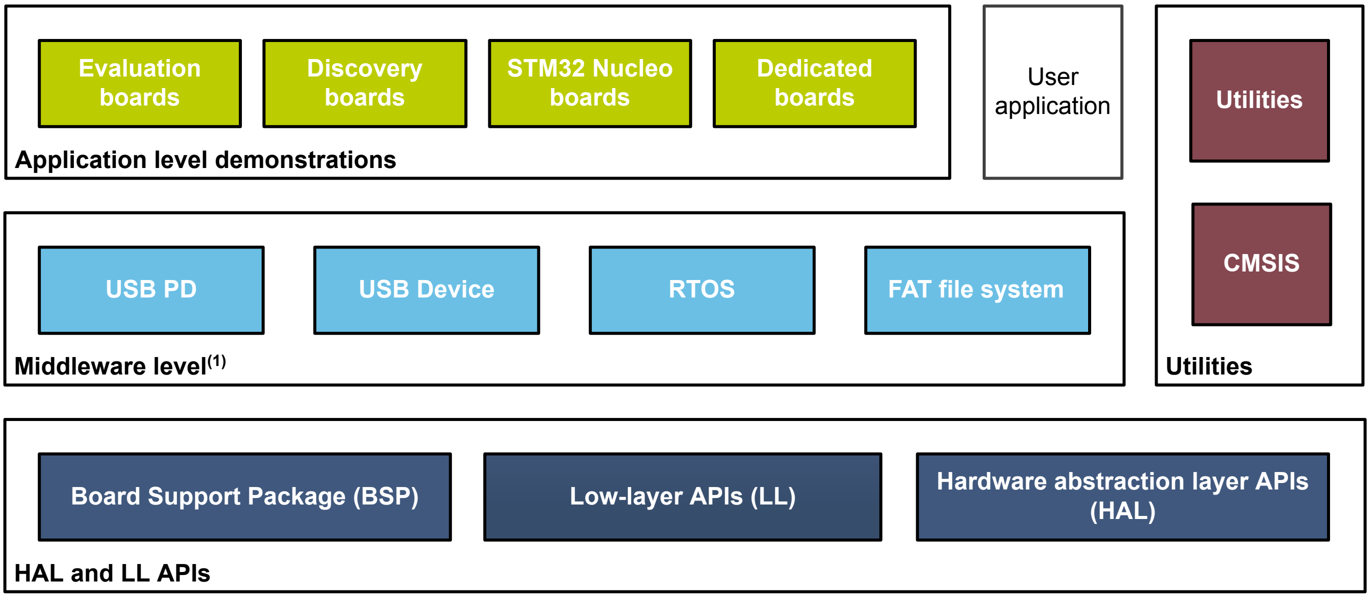
STM32Cube firmware examples for STM32G4 Series

Introduction

The STM32CubeG4 MCU Package is delivered with a 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 STM32CubeG4 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. STM32CubeG4 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. They are available on www.st.com/stm32cubefw.

- Latest release of the STM32CubeG4 MCU Package for the 32-bit microcontrollers in the STM32G4 Series based on the Arm® Cortex®-M4 processor
- *Getting started with STM32CubeG4 for STM32G4 Series* (UM2492)
- *Description of STM32G4 HAL and low-layer drivers* (UM2570)
- *STM32CubeG4 Nucleo demonstration firmware* (UM2573)
- *STM32CubeG4 STM32G474E-EVAL demonstration firmware* (UM2583)
- *Managing USB power delivery systems with STM32 microcontrollers* (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 STM32CubeG4 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, such as 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 synchroniation from TIM6 and DMA). The usage of the board resources is reduced to the strict minimum.
- **Examples_LL**
  These examples use only the LL drivers (HAL drivers and middleware components not used). They offer an optimum implementation of typical use cases of the peripheral features and configuration sequences. The examples are organized per peripheral (one folder for each peripheral, such as TIM) and 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, such as 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, such as USB Host) or by product feature that require high-level firmware bricks (such as Audio). The integration of applications that use several middleware stacks is also supported.
- **Demonstrations**
  The demonstrations aim at integrating and running the maximum number of peripherals and middleware stacks to showcase the product features and performance.
- **Template project**
  The template project is provided to allow the user to quickly build a firmware application using HAL and BSP drivers on a given board.
- **Template_LL project**
  The template LL projects are provided to allow the user to quickly build a firmware application using LL drivers on a given board.

The examples are located under `STM32Cube_FW_G4_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, and others). The BSP is based on a modular architecture that can be easily ported to any hardware by implementing the low-level routines.

Table 1. STM32CubeG4 firmware examples contains the list of examples provided with the STM32CubeG4 MCU Package.

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

Table 1. STM32CubeG4 firmware examples

Level	Module Name	Project Name	Description	B-G474E-DPOW1[(1)]	NUCLEO-G431RB[(1)]	STM32G474E-EVAL[(1)]	NUCLEO-G431KB[(1)]	NUCLEO-G474RE[(1)]
Templates	-	Starter project	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application.	X	X	X	X	X
	Total number of templates: 5			1	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.	X	X	X	X	X
	Total number of templates_ll: 5			1	1	1	1	1
Examples	-	BSP	This example provides a short description of how to use the BSP to interface with the EVAL board At the beginning of the main program the HAL_Init() function is called to reset all the peripherals, initialize the Flash interface and the systick.	X	-	X	-	-
	ADC	ADC_ContinuousConversion_TriggerSW	This example provides a short description of how to use the ADC peripheral to perform conversions in continuous mode.	-	-	MX	-	-
		ADC_GainCompensation	Use ADC Gain compensation feature to get directly voltage in mVolt from conversion without need of data post computing.	MX	MX	MX	-	MX
		ADC_GroupsRegularInjected	Use ADC to perform conversions using the two ADC groups: regular group for ADC conversion on main stream and injected group for ADC conversions limited on specific events (conversions injected within main conversions stream).	-	-	MX	-	-
		ADC_OffsetCompensation	Use ADC Offset compensation feature to translate directly conversion result from the ADC range to an application specific range without need of post computing.	MX	MX	MX	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1(1)	NUCLEO-G431RB(1)	STM32G474E-EVAL(1)	NUCLEO-G431KB(1)	NUCLEO-G474RE(1)
Examples (Cont'd)	COMP	COMP_CompareGpioVsDacInt_OutputGpio	This example shows how to configure the COMP peripheral to compare the external voltage applied on a specific pin with a sawtooth signal generated by a DAC.	MX	MX	MX	-	MX
		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	-	-
		COMP_CompareGpioVsVrefInt_OutputGpio	This example shows how to configure the COMP peripheral to compare the external voltage applied on a specific pin with an internal reference.	-	-	MX	-	-
		COMP_OutputBlanking	How to use the comparator-peripheral output blanking feature. The purpose of the output blanking feature in motor control is to prevent tripping of the current regulation upon short current spikes at the beginning of the PWM period.	MX	-	MX	-	MX
	CORDIC	CORDIC_SinCos_DMA_Perf	How to use the CORDIC peripheral to calculate sines and cosines array in DMA mode.	-	MX	MX	-	MX
		CORDIC_Sin_DMA	How to use the CORDIC peripheral to calculate array of sines in DMA mode.	-	MX	MX	MX	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	-	-	-	-
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	MX	-	-	-	-

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	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
		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
		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
		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
	CRYP	CRYP_DMA	How to use the AES peripheral to encrypt and decrypt data using AES 128 Algorithm with ECB chaining mode in DMA 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
		CORTEXM_ModePrivilege	How to modify the Thread mode privilege access and stack. Thread mode is entered on reset or when returning from an exception.	-	MX	-	-	MX
		CORTEXM_ProcessStack	How to modify the Thread mode stack. Thread mode is entered on reset, and can be entered as a result of an exception return.	-	MX	-	-	MX
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	-	MX	-	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[(1)]	NUCLEO-G431RB[(1)]	STM32G474E-EVAL[(1)]	NUCLEO-G431KB[(1)]	NUCLEO-G474RE[(1)]
Examples (Cont'd)	DAC	DAC_DMADoubleDataMode	Use DAC DMA double data mode to save AHB bandwidth and to be able to output 2 different 250kHz sine wave sampled at 15MSps by 2 different DAC converters.	-	-	MX	-	-
		DAC_DualConversion	Use DAC dual channel mode to generate signal on both DAC channels at the same time.	-	-	MX	-	-
		DAC_DualConversionFromDMA	Use DAC dual channel mode with DMA to generate signal on both DAC channels at the same time.	MX	-	MX	-	-
		DAC_SignalsGeneration2	Use the DAC peripheral to generate several signals using the DMA controller and the DAC internal wave generator.	-	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 HAL API.	-	MX	MX	-	MX
		DMA_MUXSYNC	How to use the DMA with the DMAMUX to synchronize a transfer with the LPTIM1 output signal. USART1 is used in DMA synchronized mode to send a countdown from 10 to 00, with a period of 2 seconds.	-	-	-	-	MX
	FDCAN	FDCAN_Classic_Frame_Networking	How to configure the FDCAN peripheral to send and receive Classic CAN frames.	-	-	MX	-	-
		FDCAN_Com_IT	How to achieve Interrupt Process Communication between two FDCAN units.	-	-	MX	-	-
		FDCAN_Com_polling	How to achieve Polling Process Communication between two FDCAN units.	-	-	MX	-	-
		FDCAN_Loopback	How to configure the FDCAN to operate in loopback mode.	-	-	MX	-	-

Level	Module Name	Project Name	Description	B-G474E-DPOW1 (1)	NUCLEO-G431RB (1)	STM32G474E-EVAL (1)	NUCLEO-G431KB (1)	NUCLEO-G474RE (1)
Examples (Cont'd)	FLASH	FLASH_DualBoot	This example guides you through the different configuration steps by mean of HAL API how to program bank1 and bank2 of the STM32G4xx internal Flash memory mounted on STM32G474E-EVAL Rev B and swap between both of them.	-	-	MX	-	MX
		FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal Flash memory.	-	MX	MX	MX	MX
		FLASH_FastProgram	How to configure and use the FLASH HAL API to erase and fast program the internal Flash memory.	-	MX	MX	MX	MX
		FLASH_WriteProtection	How to configure and use the FLASH HAL API to enable and disable the write protection of the internal Flash memory.	-	MX	MX	MX	MX
	FMAC	FMAC_Adaptive_FIR_AN5305	How to use the FMAC peripheral to implement an adaptive FIR filter in DMA mode.	MX	-	-	-	MX
		FMAC_Buck_VoltageMode_HW_AN5305	How to use the FMAC peripheral to implement a 3p3z controller.	MX	-	-	-	-
		FMAC_FIR_DMAToIT	How to use the FMAC peripheral to perform a FIR filter from DMA mode to IT mode.	-	MX	-	-	MX
		FMAC_FIR_PollingToIT	How to use the FMAC peripheral to perform a FIR filter from polling mode to IT mode.	-	MX	MX	-	MX
		FMAC_IIR_ITToPolling	How to use the FMAC peripheral to perform an IIR filter from IT mode to polling mode.	MX	MX	-	-	MX
		FMAC_IIR_PollingToDMA	How to use the FMAC peripheral to perform an IIR filter from polling mode to DMA mode.	MX	MX	-	MX	MX
	FMC	FMC_SRAM	This example describes how to configure the FMC controller to access the SRAM memory.	-	-	MX	-	-

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	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_TIM	How to customize HAL using a general-purpose timer as main source of time base instead of Systick.	-	MX	MX	-	MX
	HRTIM	HRTIM_Basic_ArbitraryWaveform	This example describes how to generate basic non-PWM waveforms with the HRTIM, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX
		HRTIM_Basic_MultiplePWM	This example describes how to generate basic PWM waveforms PWM on multiple outputs with the HRTIM, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX
		HRTIM_Basic_PWMMaster	This example describes how to generate basic PWM waveforms with HRTIM timers other than the timing unit itself, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX
		HRTIM_Basic_SinglePWM	This example describes how to check HRTIM outputs and to generate elementary PWM waveforms with the HRTIM, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	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
		I2C_TwoBoards_ComIT	How to handle I2C data buffer transmission/reception between two boards, using an interrupt.	-	MX	-	-	MX
		I2C_TwoBoards_ComPolling	How to handle I2C data buffer transmission/reception between two boards, in polling mode.	-	MX	-	-	MX
		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
		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
	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	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	-	-	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

Level	Module Name	Project Name	Description	B-G474E-DPOW1[(1)]	NUCLEO-G431RB[(1)]	STM32G474E-EVAL[(1)]	NUCLEO-G431KB[(1)]	NUCLEO-G474RE[(1)]
Examples (Cont'd)	OPAMP	OPAMP_Calibration	This example shows how to calibrate the OPAMP.	-	-	MX	-	-
		OPAMP_InternalFollower	This example provides a short description of how to configure the OPAMP in internal follower mode (unity gain). The signal applied on OPAMP non-inverting input is reproduced on OPAMP output.	-	-	MX	-	-
		OPAMP_PGA	This example shows how to use the built-in PGA mode (OPAMP programmable gain).	-	MX	MX	-	MX
		OPAMP_PGA_ExternalBias	This example is configuring OPAMP1 as follow: - Inverting input: PA3 (pin 42 on connector CN6) - Non inverting input: PA7 (pin 37 on connector CN6) - Output: PA2 (pin 43 on connector CN6) - Gain: Gp=2 / Gm=1 or Gp=4 / Gm=3 (User can change from one to the other using User push-button) This example also provides signals to connect to the OPAMP inputs: - A sine wave generated by DAC1 (PA4 - pin 41 on connector CN6) - A bias generated by potentiometer RV2 (pin 1 - Jumper JP5) Positive gain configuration: - Sine wave (PA4 - pin 41 on connector CN6) is connected to OPAMP's non inverting input (PA7 - pin 37 on connector CN6) - Bias (pin 1 - Jumper JP5) is connected to OPAMP's inverting input (PA3 - pin 42 on connector CN6) - OPAMP output signal is: OPAMP OUT = 2 * Sine Wave - 1 * Bias (or 4 * Sine Wave - 3 * Bias) Negative gain configuration: - Bias (pin 1 - Jumper JP5) is connected to OPAMP's non inverting input (PA7 - pin 37 on connector CN6) - Sine wave (PA4 - pin 41 on connector CN6) is connected to OPAMP's inverting input (PA3 - pin 42 on connector CN6) - OPAMP output signal is: OPAMP OUT = 2 * Bias - 1 * Sine Wave (or 4 * Bias - 3 * Sine Wave) - Connection needed: - Connect an oscilloscope to each OPAMP pin in order to observe this example behavior: PA3 (pin 42 on connector CN6) PA7 (pin 37 on connector CN6) PA2 (pin 43 on connector CN6) - Connect sine wave and bias to the OPAMP inputs and make the bias vary to observe OPAMP behavior evolution.	-	-	MX	-	-
		OPAMP_TimerControlMux	This mode allows upon a timer trigger to change OPAMP configuration from a primary one to a secondary one. Possibilities are as follow: Primary configuration is standalone: - Secondary is standalone with possibility to change either one or both inputs Primary configuration is follower or PGA: - Secondary can be follower with same or different non inverting input - Secondary can be PGA with same or different non inverting input This example is configuring OPAMP4 as follow: - Primary configuration is follower with non inverting input on DAC4 generating a triangle wave.	-	-	MX	-	-

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	PWR	PWR_CurrentConsumption	How to configure the system to measure the current consumption in different low-power modes.	-	MX	-	-	MX
		PWR_LPRUN	How to enter and exit the Low-power run mode.	-	MX	-	-	MX
		PWR_LPRUN_SRAM1	This example shows how to enter and exit the Low Power Run mode.	-	MX	MX	-	-
		PWR_LPSLEEP	How to enter the Low-power sleep mode and wake up from this mode by using an interrupt.	-	MX	-	-	MX
		PWR_PVD	How to configure the programmable voltage detector by using an external interrupt line. External DC supply must be used to supply Vdd.	-	-	MX	-	MX
		PWR_SHUTDOWN	This example shows how to enter the system in SHUTDOWN mode and wake-up from this mode using external RESET or WKUP pin.	-	MX	-	-	MX
		PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	-	MX	-	-	-
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using an external reset or the WKUP pin.	-	MX	-	-	-
		PWR_STANDBY_RTC	How to enter the Standby mode and wake-up from this mode by using an external reset or the RTC wakeup timer.	-	-	-	MX	MX
		PWR_STOP0	This example shows how to enter Stop 0 mode and wake up from this mode using an interrupt.	-	MX	-	-	MX
		PWR_STOP0_RTC	This example shows how to enter Stop 0 mode and wake up from this mode using an interrupt from RTC Wake-up Timer.	-	-	MX	MX	MX
		PWR_STOP1	This example shows how to enter Stop 1 mode and wake up from this mode using an interrupt.	-	MX	-	-	-
		PWR_STOP1_RTC	This example shows how to enter Stop 1 mode and wake up from this mode using an interrupt from RTC Wake-up Timer.	-	-	-	MX	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	QSPI	QSPI_ExecuteInPlace	This example describes how to execute a part of the code from the QSPI memory. To do this, a section is created where the function is stored.	-	-	MX	-	-
		QSPI_MemoryMapped	This example describes how to erase part of the QSPI memory, write data in DMA mode and access to QSPI memory in memory-mapped mode to check the data in a forever loop.	-	-	MX	-	-
		QSPI_MemoryMappedDual	This example describes how to use QSPI interface in memory mapped dual flash mode.	-	-	MX	-	-
		QSPI_ReadWriteDual_DMA	This example describes how to use QSPI interface in dual flash mode.	-	-	MX	-	-
		QSPI_ReadWrite_DMA	This example describes how to erase part of the QSPI memory, write data in DMA mode, read data in DMA mode and compare the result in a forever loop.	-	-	MX	-	-
		QSPI_ReadWrite_IT	This example describes how to erase part of the QSPI memory, write data in IT mode, read data in IT mode and compare the result in a forever loop.	-	-	MX	-	-
	RCC	RCC_CRS_Synchronization_IT	Configuration of the clock recovery service (CRS) in Interrupt mode, using the RCC HAL API.	-	-	-	-	MX
		RCC_CRS_Synchronization_Polling	Configuration of the clock recovery service (CRS) in Polling mode, using the RCC HAL API.	-	MX	MX	-	MX
		RCC_ClockConfig	Configuration of the system clock (SYSCLK) and modification of the clock settings in Run mode, using the RCC HAL API.	-	-	-	-	MX
	RNG	RNG_MultiRNG	Configuration of the RNG using the HAL API. This example uses the RNG to generate 32-bit long random numbers.	-	MX	MX	-	MX
		RNG_MultiRNG_IT	Configuration of the RNG using the HAL API. This example uses RNG interrupts to generate 32-bit long random numbers.	-	MX	MX	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	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_LSI	Use of the LSI clock source autocalibration to get a precise RTC clock.	-	MX	MX	-	MX
	SAI	SAI_AudioPlay	This example shows how to use the SAI HAL API to play an audio file using the DMA circular mode and how to handle the buffer update.	-	-	MX	-	-
	SMART CARD	SMARTCARD_T0_MFX	This example describes a firmware smartcard Interface based on USART.	-	-	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
		SPI_FullDuplex_ComIT_Slave	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	-	MX	-	-	MX
		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

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	TIM	TIM_CascadeSynchro	This example shows how to synchronize TIM2 and Timers (TIM3 and TIM4) in cascade mode.	-	MX	MX	MX	MX
		TIM_Combined	This example shows how to configure the TIM1 peripheral to generate 3 PWM combined signals with TIM1 Channel5.	-	-	-	-	MX
		TIM_ComplementarySignals	This example shows how to configure the TIM1 peripheral to generate three complementary TIM1 signals, to insert a defined dead time value, to use the break feature and to lock the desired parameters.	-	MX	MX	-	MX
		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	MX
		TIM_DMABurst	How to update the TIMER channel 1 period and duty cycle using the TIMER DMA burst feature.	-	-	-	-	MX
		TIM_Dithering	This example shows how to configure the TIM3 peripheral in PWM mode with dithering.	-	MX	MX	-	MX
		TIM_Encoder	This example shows how to configure the TIM1 peripheral in encoder mode to determinate the rotation direction.	-	-	-	-	MX
		TIM_EncoderIndex_PulseOnCompare	This example shows how to configure the TIM3 peripheral in encoder mode with index and generate a pulse on a certain value of encoder interface counter with pulse on compare.	-	-	-	-	MX
		TIM_InputCapture	How to use the TIM peripheral to measure an external signal frequency.	-	MX	MX	MX	MX
		TIM_OCToggle	Configuration of the TIM peripheral to generate four different signals at four different frequencies.	-	-	-	-	MX
		TIM_OnePulse	This example shows how to use the TIMER peripheral to generate a single pulse when a rising edge of an external signal is received on the TIMER Input pin.	-	-	MX	-	-
		TIM_PWMInput	How to use the TIM peripheral to measure the frequency and duty cycle of an external signal.	MX	MX	MX	MX	MX
		TIM_PWMOutput	This example shows how to configure the TIM peripheral in PWM (Pulse Width Modulation) mode.	MX	MX	MX	MX	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples (Cont'd)	UART	LPUART_TwoBoards_ComIT	LPUART transmission (transmit/receive) in Interrupt mode between two boards.	-	MX	-	-	MX
		LPUART_WakeUpFromStop	Configuration of an LPUART to wake up the MCU from Stop mode when a given stimulus is received.	-	MX	MX	-	MX
		UART_HyperTerminal_DMA	UART transmission (transmit/receive) in DMA mode between a board and an HyperTerminal PC application.	-	MX	MX	MX	MX
		UART_HyperTerminal_IT	UART transmission (transmit/receive) in Interrupt mode between a board and an HyperTerminal PC application.	-	MX	MX	-	MX
		UART_Printf	Re-routing of the C library printf function to the UART.	MX	MX	MX	-	MX
		UART_TwoBoards_ComDMA	UART transmission (transmit/receive) in DMA mode between two boards.	-	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
		UART_WakeUpFromStopUsingFIFO	This example shows how to use UART HAL API to wake up the MCU from STOP mode using the UART FIFO level.	-	MX	MX	-	MX
	USART	USART_SlaveMode	This example describes an USART-SPI communication (transmit/receive) between two boards where the USART is configured as a slave.	-	MX	-	-	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	MX	MX	MX
	Total number of examples: 278			16	77	72	21	92

Level	Module Name	Project Name	Description	B-G474E-DPOW1[(1)]	NUCLEO-G431RB[(1)]	STM32G474E-EVAL[(1)]	NUCLEO-G431KB[(1)]	NUCLEO-G474RE[(1)]
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_Init	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	-	MX	-	-	MX
		ADC_GroupsRegularInjected_Init	How to use an ADC peripheral with both ADC groups (regular and injected) in their intended use cases.	-	MX	-	-	MX
		ADC_Oversampling_Init	How to use an ADC peripheral with ADC oversampling.	-	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
		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
	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 STM32G4xx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-	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 STM32G4xx COMP LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	MX	-	-	MX
	CORDIC	CORDIC_CosSin	How to use the CORDIC peripheral to calculate cosine and sine.	-	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.	-	X	-	-	X

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples_LL (Cont'd)	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
	CRS	CRS_Synchronization_IT	How to configure the clock recovery service in IT mode through the STM32G4xx CRS LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	MX
		CRS_Synchronization_Polling	How to configure the clock recovery service in polling mode through the STM32G4xx CRS LL API. The peripheral initialization uses 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 STM32G4xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	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 STM32G4xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	MX
		DAC_GenerateWaveform_TriggerHW_Init	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 STM32G4xx DAC LL API. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	-	-	-	-	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

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples_LL (Cont'd)	EXTI	EXTI_ToggleLedOnIT	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 STM32G4xx LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	MX
		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 STM32G4xx 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 STM32G4xx 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 STM32G4xx LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	-	MX	-	-	MX
	HRTIM	HRTIM_Basic_Arbitrary_Waveform	This example describes how to generate basic non-PWM waveforms with the HRTIM, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX
		HRTIM_Basic_Multiple_PWM	This example describes how to generate basic PWM waveforms PWM on multiple outputs with the HRTIM, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX
		HRTIM_Basic_PWM_Master	This example describes how to generate basic PWM waveforms with HRTIM timers other than the timing unit itself, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX
		HRTIM_Basic_Single_PWM	This example describes how to check HRTIM outputs and to generate elementary PWM waveforms with the HRTIM, as per HRTIM Cookbook basic examples (refer to AN4539 Application note).	-	-	-	-	MX
		HRTIM_CBC_Deadtime	This example describes how to implement a cycle-by-cycle (CBC) current control with complementary signals and deadtime insertion.	-	-	-	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples_LL (Cont'd)	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	-	-	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	-	-	MX
		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_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	-	-	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
		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	-	-	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
		I2C_TwoBoards_WakeUpFromStop_IT_Init	How to handle the reception of a data byte from an I2C slave device in Stop 1 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

Level	Module Name	Project Name	Description	B-G474E-DPOW1(1)	NUCLEO-G431RB(1)	STM32G474E-EVAL(1)	NUCLEO-G431KB(1)	NUCLEO-G474RE(1)
Examples_LL (Cont'd)	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 STM32G4xx LPTIM LL API. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	X	-	-	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 STM32G4xx LPTIM LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	-	MX	-	-	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).	-	MX	-	-	MX
		LPUART_WakeUpFromStop_Init	Configuration of GPIO and LPUART peripherals to allow characters received on LPUART_RX pin to wake up the MCU from low-power mode. This example is based on the LPUART LL API. The peripheral initialization uses LL initialization function to demonstrate LL init usage.	-	MX	-	-	MX
	OPAMP	OPAMP_Follower	How to use the OPAMP peripheral in follower mode. To test OPAMP in this example, a voltage waveform is generated by the DAC peripheral and can be connected to OPAMP input. This example is based on the STM32G4xx OPAMP LL API. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	X	-	-	X
		OPAMP_PGA	How to use the OPAMP peripheral in PGA mode (programmable gain amplifier). To test OPAMP, a voltage waveform is generated by the DAC and feeds the OPAMP input. This example is based on the STM32G4xx OPAMP LL API. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	X	-	-	X

Level	Module Name	Project Name	Description	B-G474E-DPOW1(1)	NUCLEO-G431RB(1)	STM32G474E-EVAL(1)	NUCLEO-G431KB(1)	NUCLEO-G474RE(1)
Examples_LL (Cont'd)	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
		PWR_EnterStopMode	How to enter the Stop 1 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
	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	-	-	MX
		RNG_GenerateRandomNumbers_IT	Configuration of the RNG to generate 32-bit long random numbers using interrupts. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples_LL (Cont'd)	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_ProgrammingTheWakeUpTimer	Configuration of the RTC to use the WUT. 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_DMA	Configuration of GPIO and SPI peripherals to transmit bytes from a SPI Master device to a SPI Slave device in DMA mode. This example is based on the STM32G4xx 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 STM32G4xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	MX
		SPI_TwoBoards_FullDuplex_DMA_Master_Init	Data buffer transmission and receptionvia SPI using DMA mode. This example is based on the STM32G4xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	MX
		SPI_TwoBoards_FullDuplex_DMA_Slave_Init	Data buffer transmission and receptionvia SPI using DMA mode. This example is based on the STM32G4xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	MX	-	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Examples_LL (Cont'd)	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	-	-	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 STM32G4xx 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 STM32G4xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	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 STM32G4xx 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 STM32G4xx 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 STM32G4xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-	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 STM32G4xx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL Init.	-	MX	-	-	MX

Level	Module Name	Project Name	Description	B-G474E-DPOW1 (1)	NUCLEO-G431RB (1)	STM32G474E-EVAL (1)	NUCLEO-G431KB (1)	NUCLEO-G474RE (1)
Examples_LL (Cont'd)	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_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_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 STM32G4xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	MX	-	-	MX
		USART_Communication_Tx_IT_Init	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on STM32G4xx 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 STM32G4xx USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purpose (performance and size).	-	MX	-	-	MX
		USART_HardwareFlowControl	Configuration of GPIO and USART1 peripheral to receive characters asynchronously from an HyperTerminal (PC) in Interrupt mode with the Hardware Flow Control feature enabled. This example is based on STM32G4xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-	X

Level	Module Name	Project Name	Description	B-G474E-DPOW1[(1)]	NUCLEO-G431RB[(1)]	STM32G474E-EVAL[(1)]	NUCLEO-G431KB[(1)]	NUCLEO-G474RE[(1)]
Examples_LL (Cont'd)	USART (Cont'd)	USART_SyncCommunication_FullDuplex_DMA	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 STM32G4xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-	X
		USART_SyncCommunication_FullDuplex_IT	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 STM32G4xx 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).	-	X	-	-	X
		USART_WakeUpFromStop1	Configuration of GPIO and USART peripherals to receive characters on USART_RX pin and wake up the MCU from low-power mode. This example is based on the STM32G4xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-	X
		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: 154			0	74	0	0	80

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[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 STM32G4xx ADC HAL and LL API. The LL API is used for performance improvement.	-	MX	-	-	MX
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the STM32G4xx DMA HAL and LL API. The LL API is used for performance improvement.	-	MX	-	-	MX
	HRTIM	HRTIM_Buck_Boost	This example shows how to configure the HRTIM to control a non-inverting buck-boost converter timer.	MX	-	-	-	-
		HRTIM_Buck_Sync_Rect	This example shows how to configure the HRTIM to control a buck converter with synchronous rectification.	MX	-	-	-	-
		HRTIM_Dual_Buck	This example shows how to configure the HRTIM to have 2 buck converters controlled by a single timer unit.	MX	-	-	-	-
	PWR	PWR_STOP1	How to enter the STOP 1 mode and wake up from this mode by using external reset or wakeup interrupt (all the RCC function calls use RCC LL API for minimizing footprint and maximizing performance).	-	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 STM32G4xx 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 STM32G4xx UART HAL and LL API, the LL API being used for performance improvement.	-	MX	-	-	MX
	Total number of examples_mix: 13			3	5	0	0	5

Level	Module Name	Project Name	Description	B-G474E-DPOW1[1]	NUCLEO-G431RB[1]	STM32G474E-EVAL[1]	NUCLEO-G431KB[1]	NUCLEO-G474RE[1]
Applications	FatFs	FatFs_RAMDisk	This application provides a description on how to use STM32Cube firmware with FatFs middleware component as a generic FAT file system module, in order to develop an application exploiting FatFs offered features with RAM disk (SDRAM) drive configuration.	-	-	MX	-	-
		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	-	-
		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	-	-
		FreeRTOS_SemaphoreFromISR	How to use semaphore from ISR with CMSIS RTOS API.	-	-	MX	-	-
		FreeRTOS_Signal	How to perform thread signaling using CMSIS RTOS API.	-	-	MX	-	-
		FreeRTOS_SignalFromISR	This application shows the usage of CMSIS-OS Signal API from ISR context.	-	-	MX	-	-
		FreeRTOS_ThreadCreation	How to implement thread creation using CMSIS RTOS API.	-	-	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	-	MX	-	-
		USB-PD_Provider_1port	How to create a simple type C provider.	-	-	MX	-	-

Level	Module Name	Project Name	Description	B-G474E-DPOW1(1)	NUCLEO-G431RB(1)	STM32G474E-EVAL(1)	NUCLEO-G431KB(1)	NUCLEO-G474RE(1)
	USB_Device	CDC_Standalone	This application describes how to use USB device application based on the Device Communication Class (CDC) following the PSTN sub-protocol on the STM32G4xx devices.	-	-	MX	-	-
		DFU_Standalone	Compliant implementation of the Device Firmware Upgrade (DFU) capability to program the embedded Flash memory through the USB peripheral.	-	-	MX	-	-
		HID_Standalone	Use of the USB device application based on the Human Interface (HID).	-	-	MX	-	-
		MSC_Standalone	This application shows how to use the USB device application based on the Mass Storage Class (MSC) on the STM32G4xx devices.	-	-	MX	-	-
	Total number of applications: 27			1	4	17	1	4
Demonstrations	-	Adafruit_LCD_1_8_SD_Joystick	This demonstration provides a short description of how to use the BSP drivers.	-	X	-	-	X
		Binary		-	-	X	-	-
		Demo	This demonstration firmware is based on STM32Cube. It helps you to discover STM32 Cortex-M devices that can be plugged on a STM32 Discovery board.	-	-	X	-	-
		Led_Jumper	This demonstration provides a short description of how to use the BSP drivers.	-	-	-	X	-
	Total number of demonstrations: 5			0	1	2	1	1
Total number of projects: 487				22	163	93	25	184

1. *STM32CubeMX-generated examples are highlighted with the MX STM32CubeMX icon. Other examples are marked with an "X".*

Revision history

Table 2. Document revision history

Date	Version	Changes
14-May-2019	1	Initial release.
05-Sep-2019	2	Updated Section 1 Reference documents Modified Table 1. STM32CubeG4 firmware examples.

Contents

List of tables