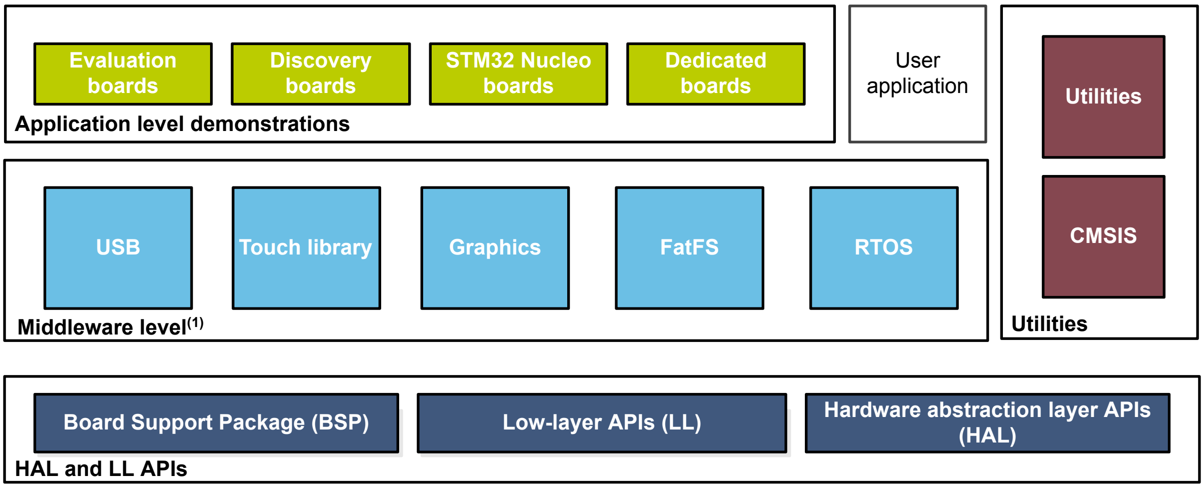
STM32Cube firmware examples for STM32L1 Series

Introduction

The STM32CubeL1 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).

Figure 1. STM32CubeL1 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 STM32CubeL1 MCU Package for the 32-bit microcontrollers in the STM32L1 Series based on the Arm® Cortex®-M processor
- *Getting started with STM32CubeL1 for STM32L1 Series* (UM1802)
- *STM32CubeL1 Nucleo demonstration firmware* (UM1804)
- *Description of STM32L1 HAL and low-layer drivers* (UM1816)
- *STM32Cube USB device library* (UM1734)
- *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 STM32CubeL1 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 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, 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_L1_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

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 contains the list of examples provided with the STM32CubeL1 MCU Package.

Table 1. STM32CubeL1 firmware examples

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Templates	-	Starter project	This project provides a reference template that can be used to build any firmware application.	X	X	X	X
	Total number of templates: 4			1	1	1	1
Templates_LL	-	Starter project	This project provides a reference template through the LL API that can be used to build any firmware application.	X	X	X	X
	Total number of templates_ll: 4			1	1	1	1
Examples	-	BSP	This example provides a description of how to use the different BSP drivers.	-	-	-	X
	ADC	ADC_AnalogWatchdog	How to use the ADC peripheral to perform conversions with an analog watchdog and out-of-window interrupts enabled.	X	-	-	-
		ADC_LowPower	How to use the ADC peripheral to perform conversions with ADC low-power modes: auto-wait and auto-power off.	-	-	-	X
		ADC_Regular_injected_groups	How to use the ADC peripheral to perform conversions using the two ADC groups: regular group for ADC conversions on the main stream, and injected group for ADC conversions limited to specific events (conversions injected into the main conversion stream).	-	-	X	-
		ADC_Sequencer	How to use the ADC peripheral with a sequencer to convert several channels	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples	COMP	COMP_AnalogWatchdog	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 (V_{REFINT}) and a fraction of the internal voltage reference ($V_{REFINT}/4$), in interrupt mode.	-	-	-	X
		COMP_Interrupt	How to use a comparator peripheral to compare a voltage level applied on a GPIO pin to the the internal voltage reference (V_{REFINT}), in interrupt mode. When the comparator input crosses (either rising or falling edges) the internal reference voltage V_{REFINT} (1.22V), the comparator generates an interrupt.	X	-	-	-
		COMP_PWMSignalControl	How to configure a comparator peripheral to automatically hold the TIMER PWM output in the safe state (low level) as soon as the comparator output is set to a high level.	-	X	-	-
		COMP_PulseWidthMeasurement	How to configure a comparator peripheral to measure pulse width. This method (measuring signal pulses using a comparator) is useful when an external signal does not respect the VIL and VIH levels	-	-	X	-
	CRC	CRC_Example	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes the CRC code of a given buffer of 32-bit data words, using a fixed generator polynomial (0x4C11DB7).	-	-	-	X
	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.	-	-	-	X
		CORTEXM_ModePrivilege	How to modify the Thread mode privilege access and stack. Thread mode is entered on reset or when returning from an exception	-	-	-	X
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs	-	-	-	X
	DAC	DAC_SignalsGeneration	How to use the DAC peripheral to generate several signals using the DMA controller.	-	-	-	X
		DAC_SimpleConversion	How to use the DAC peripheral to do a simple conversion.	-	X	-	-
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the HAL API.	-	-	-	X
	FLASH	FLASH_EraseProgram	How to configure and use the FLASH HAL API to erase and program the internal Flash memory.	-	-	-	X
		FLASH_WriteProtection	How to configure and use the FLASH HAL API to enable and disable the write protection of the internal Flash memory.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples	FSMC	FSMC_NOR	How to configure the FSMC controller to access the NOR memory	-	-	-	X
		FSMC_SRAM	How to configure the FSMC controller to access the SRAM memory.	-	-	-	X
		FSMC_SRAM_DataMemory	How to configure the FSMC controller to access the SRAM memory including the heap and stack.	-	-	-	X
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	-	X	-	-
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	X	X	X	X
	HAL	HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as main source of time base, instead of Systick.	-	-	-	X
	I2C	I2C_EEPROM	How to handle I2C data buffer transmission/ reception with DMA. In the example, the device communicates with an I2C EEPROM memory.	-	-	-	X
		I2C_TwoBoards_AdvComIT	How to handle I2C data buffer transmission/ reception between two boards, using an interrupt.	-	X	-	-
		I2C_TwoBoards_ComDMA	How to handle I2C data buffer transmission/ reception between two boards, via DMA.	-	X	-	-
		I2C_TwoBoards_ComIT	How to handle I2C data buffer transmission/ reception between two boards, using an interrupt.	-	X	-	-
		I2C_TwoBoards_ComPolling	How to handle I2C data buffer transmission/ reception between two boards, in polling mode.	-	X	-	-
		I2C_TwoBoards_RestartAdvComIT	How to perform multiple I2C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	-	X	-	-
		I2C_TwoBoards_RestartComIT	How to handle single I2C data buffer transmission/ reception between two boards, in interrupt mode and with restart condition.	-	X	-	-
	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.	-	-	X	-
	LCD	LCD_Blink_Frequency	How to use the embedded LCD glass controller and how to set the LCD blink mode and blinking frequency.	X	-	-	-
		LCD_Contrast_Control	How to use the embedded LCD glass controller and how to set the LCD contrast.	-	-	-	X
		LCD_SegmentsDrive	How to use the embedded LCD glass controller to drive the on-board LCD glass by Pacific Display Devices.	X	-	-	-
	OPAMP	OPAMP_CALIBRATION	How to calibrate the OPAMP peripheral.	-	-	-	X
		OPAMP_InternalFollower	How to configure the OPAMP peripheral in internal follower mode (unity gain). The signal applied on OPAMP non-inverting input is reproduced on OPAMP output.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples	PWR	PWR_LPRUN	How to enter and exit the Low-power run mode.	-	X	-	-
		PWR_LPSLEEP	How to enter the Low-power sleep mode and wake up from this mode by using an interrupt.	-	X	-	-
		PWR_PVD	How to configure the programmable voltage detector by using an external interrupt line. External DC supply must be used to supply Vdd.	-	-	-	X
		PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	-	X	-	-
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using an external reset or the WKUP pin.	-	-	-	X
		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.	-	X	-	-
		PWR_STOP	How to enter the Stop mode and wake up from this mode by using the RTC wakeup timer event or an interrupt.	-	-	X	-
		PWR_STOP_RTC	How to enter the Stop mode and wake up from this mode by using the RTC wakeup timer event connected to an interrupt.	-	X	-	-
	RCC	RCC_ClockConfig	Configuration of the system clock (SYSCLK) and modification of the clock settings in Run mode, using the RCC HAL API.	X	X	X	X
	RTC	RTC_Alarm	Configuration and generation of an RTC alarm using the RTC HAL API.	-	-	X	X
		RTC_Tamper	Configuration of the RTC HAL API to write/read data to/from RTC Backup registers.	X	-	-	X
	SPI	SPI_FullDuplex_ComDMA	Data buffer transmission/reception between two boards via SPI using DMA.	-	X	-	-
		SPI_FullDuplex_ComIT	Data buffer transmission/reception between two boards via SPI using Interrupt mode.	-	X	-	-
		SPI_FullDuplex_ComPolling	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	X	-	-
	TIM	TIM_DMA	Use of the DMA with TIMER Update request to transfer data from memory to TIMER Capture Compare Register 3 (TIMx_CCR3).	-	X	-	-
		TIM_InputCapture	Use of the TIM peripheral to measure an external signal frequency.	-	-	-	X
		TIM_PWMOutput	Configuration of the TIM peripheral in PWM (pulse width modulation) mode.	-	X	-	-
		TIM_TimeBase	Configuration of the TIM peripheral to generate a timebase of one second with the corresponding interrupt request.	-	-	-	X

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples	UART	UART_HyperTerminal_DMA	UART transmission (transmit/receive) in DMA mode between a board and an HyperTerminal PC application.	-	-	-	X
		UART_Printf	Re-routing of the C library printf function to the UART.	-	-	-	X
		UART_TwoBoards_ComDMA	UART transmission (transmit/receive) in DMA mode between two boards.	-	X	-	-
		UART_TwoBoards_ComIT	UART transmission (transmit/receive) in Interrupt mode between two boards.	-	X	-	-
		UART_TwoBoards_ComPolling	UART transmission (transmit/receive) in Polling mode between two boards.	-	X	-	-
	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.	-	-	-	X
	Total number of examples: 69			7	27	7	28

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	ADC	ADC_AnalogWatchdog	How to use an ADC peripheral with an ADC analog watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds.	-	X	-	-
		ADC_ContinuousConversion_TriggerSW	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	-	X	-	-
		ADC_ContinuousConversion_TriggerSW_Init	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	-	X	-	-
		ADC_ContinuousConversion_TriggerSW_LowPower	How to use an ADC peripheral with ADC low-power features.	-	X	-	-
		ADC_GroupsRegularInjected	This example describes how to use a ADC peripheral with both ADC groups (ADC group regular and ADC group injected) in their intended use case. This example is based on the STM32L1xx ADC LL API.	-	X	-	-
		ADC_MultiChannelSingleConversion	How to use an ADC peripheral to convert several channels. ADC conversions are performed successively in a scan sequence.	-	X	-	-
		ADC_SingleConversion_TriggerSW	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).	-	X	-	-
		ADC_SingleConversion_TriggerSW_DMA	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).	-	X	-	-
		ADC_SingleConversion_TriggerSW_IT	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).	-	X	-	-
		ADC_SingleConversion_TriggerTimer_DMA	How to use an ADC peripheral to perform a single ADC conversion on a channel at each trigger event from a timer. Converted data is indefinitely transferred by DMA into a table (circular mode).	-	X	-	-
		ADC_TemperatureSensor	How to use an ADC peripheral to perform a single ADC conversion on the internal temperature sensor and calculate the temperature in degrees Celsius.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	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 (V_{REFINT}), in interrupt mode. This example is based on the STM32L1xx 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 (V_{REFINT}), in interrupt mode. This example is based on the STM32L1xx COMP LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	X	-	-
		COMP_CompareGpioVsVrefInt_Window_IT	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 (V_{REFINT}) and a fraction of the internal voltage reference ($V_{REFINT}/2$), in interrupt mode. This example is based on the STM32L1xx COMP LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	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	-	-
	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).	-	X	-	-
	DAC	DAC_GenerateConstantSignal_TriggerSW	How to use the DAC peripheral to generate a constant voltage signal. This example is based on the STM32L1xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		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 STM32L1xx DAC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size) Example configuration: One DAC channel (DAC1 channel1) is configured to connect DAC channel output on GPIO pin to get the samples from DMA transfer and to get conversion trigger from timer.	-	X	-	-
		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 STM32L1xx DAC LL API. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
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	-	-
		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.	-	X	-	-
	EXTI	EXTI_ToggleLedOnIT	How to configure the $moduleName$ and use GPIOs to toggle the user LEDs available on the board when a user button is pressed. It is based on the STM32L1xx LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		EXTI_ToggleLedOnIT_Init	How to configure the EXTI and use GPIOs to toggle the user LEDs available on the board when a user button is pressed. This example is based on the STM32L1xx LL API. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	-	X	-	-
	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 STM32L1xx GPIO LL API. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	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 STM32L1xx LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	I2C	I2C_OneBoard_AdvCommunication_DMAAndIT	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.	-	X	-	-
		I2C_OneBoard_Communication_DMAAndIT	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.	-	X	-	-
		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	-	-
		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.	-	X	-	-
		I2C_OneBoard_Communication_PollingAndIT	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.	-	X	-	-
		I2C_TwoBoards_MasterRx_SlaveTx_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	-	-
		I2C_TwoBoards_MasterTx_SlaveRx	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.	-	X	-	-
		I2C_TwoBoards_MasterTx_SlaveRx_DMA	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.	-	X	-	-
	IWDG	IWDG_RefreshUntilUserEvent	How to configure the IWDG peripheral to ensure periodical counter update and generate an MCU IWDG reset when a User Button is pressed. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	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 STM32L1xx OPAMP LL API. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	X	-	-
		OPAMP_Follower_Init	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 STM32L1xx OPAMP LL API. The peripheral is initialized with LL initialization function to demonstrate LL init usage.	-	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.	-	X	-	-
		PWR_EnterStopMode	How to enter the Stop mode.	-	X	-	-
		PWR_LPRunMode_SRAM1	How to execute code in Low-power run mode from SRAM1.	-	X	-	-
		PWR_OptimizedRunMode	How to increase/decrease frequency and VCORE and how to enter/exit the Low-power run mode.	-	X	-	-
	RCC	RCC_OutputSystemClockOnMCO	Configuration of MCO pin (PA8) to output the system clock.	-	X	-	-
		RCC_UseHSEasSystemClock	Use of the RCC LL API to start the HSE and use it as system clock.	-	X	-	-
		RCC_UseHSI_PLLasSystemClock	Modification of the PLL parameters in run time.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	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	-	-
		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.	-	X	-	-
		RTC_Calendar	Configuration of the LL API to set the RTC calendar. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		RTC_ExitStandbyWithWakeUpTimer	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).	-	X	-	-
		RTC_Tamper	Configuration of the Tamper using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		RTC_TimeStamp	Configuration of the Timestamp using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	SPI	SPI_OneBoard_HalfDuplex_DMA	Configuration of GPIO and SPI peripherals to transmit bytes from an SPI Master device to an SPI Slave device in DMA mode. This example is based on the STM32L1xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		SPI_OneBoard_HalfDuplex_DMA_Init	Configuration of GPIO and SPI peripherals to transmit bytes from an SPI Master device to an SPI Slave device in DMA mode. This example is based on the STM32L1xx SPI LL API. The peripheral initialization uses the LL initialization function to demonstrate LL init usage.	-	X	-	-
		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 STM32L1xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		SPI_TwoBoards_FullDuplex_DMA	Data buffer transmission and receptionvia $COM_INSTANCE1_TYPE$ using DMA mode. This example is based on the STM32L1xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		SPI_TwoBoards_FullDuplex_IT	Data buffer transmission and receptionvia $COM_INSTANCE1_TYPE$ using Interrupt mode. This example is based on the STM32L1xx SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	TIM	TIM_DMA	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 STM32L1xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		TIM_InputCapture	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 STM32L1xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		TIM_OnePulse	This example shows how to configure 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 STM32L1xx TIM LL API. The peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	X	-	-
		TIM_OutputCompare	Configuration of the TIM peripheral to generate an output waveform in different output compare modes. This example is based on the STM32L1xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		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 STM32L1xx 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 STM32L1xx TIM LL API. The peripheral initialization uses LL initialization function to demonstrate LL init.	-	X	-	-
		TIM_TimeBase	Configuration of the TIM peripheral to generate a timebase. This example is based on the STM32L1xx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	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	-	-
		USART_Communication_Rx_IT_Continuous	Configuration of GPIO and USART peripherals to continuously 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	-	-
		USART_Communication_Rx_IT_Init	Configuration of GPIO and USART peripherals to receive characters from an HyperTerminal (PC) in Asynchronous mode using an interrupt. The peripheral initialization uses the LL initialization function to demonstrate LL init.	-	X	-	-
		USART_Communication_Tx	Configuration of GPIO and USART peripherals to send characters asynchronously to an HyperTerminal (PC) in Polling mode. If the transfer could not be complete within the allocated time, a timeout allows to exit from the sequence with timeout error. This example is based on STM32L1xx USART LL API.	-	X	-	-
		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_Tx_IT	Configuration of GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in Interrupt mode. This example is based on the STM32L1xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-
		USART_HardwareFlowControl	Configuration of GPIO and $UART_INSTANCE$ peripheral to receive characters asynchronously from an HyperTerminal (PC) in Interrupt mode with the Hardware Flow Control feature enabled. This example is based on STM32L1xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_LL	USART (continued)	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 STM32L1xx USART LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	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 STM32L1xx 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	-	-
	UTILS	UTILS_ConfigureSystemClock	This example describes how to use UTILS LL API to configure the system clock using PLL with HSI as source clock. The user application just needs to calculate PLL parameters using STM32CubeMX and call the UTILS LL API.	-	X	-	-
		UTILS_ReadDeviceInfo	This example describes how to Read UID, Device ID and Revision ID and save them into a global information buffer.	-	X	-	-
	WWDG	WWDG_RefreshUntilUserEvent	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).	-	X	-	-
	Total number of examples_ll: 73			0	73	0	0

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_MIX	ADC	ADC_SingleConversion_TriggerSW_IT	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).	-	X	-	-
	CRC	CRC_CalculateAndCheck	How to use a CRC peripheral through the STM32L1xx CRC HAL and LL API (an LL API is used for performance improvement). A fixed CRC-32 (Ethernet) generator polynomial: 0x4C11DB7 is used in the CRC peripheral.	-	X	-	-
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from Flash memory to embedded SRAM through the STM32L1xx DMA HAL and LL API. The LL API is used for performance improvement.	-	X	-	-
	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) and different maximum speed (400 kHz or 100 kHz). This example uses the STM32L1xx I2C HAL and LL API (LL API usage for performance improvement) and an interrupt.	-	X	-	-
	OPAMP	OPAMP_CALIBRATION	How to calibrate and operate the OPAMP peripheral. This example is based on the STM32L1xx OPAMP HAL and LL API (the latter to maximize performance).	-	X	-	-
	PWR	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 through the STM32L1xx RTC and RCC HAL, and LL API (LL API use for maximizing performance).	-	X	-	-
		PWR_STOP	How to enter the Stop 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).	-	X	-	-
	SPI	SPI_FullDuplex_ComPolling	Data buffer transmission/reception between two boards via SPI using Polling mode.	-	X	-	-
		SPI_HalfDuplex_ComPollingIT	Data buffer transmission/reception between two boards via SPI using Polling (LL driver) and Interrupt modes (HAL driver).	-	X	-	-
	TIM	TIM_PWMInput	Use of the TIM peripheral to measure an external signal frequency and duty cycle.	-	X	-	-

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Examples_MIX	UART	UART_HyperTerminal_IT	Use of a UART to transmit data (transmit/receive) between a board and an HyperTerminal PC application in Interrupt mode. This example describes how to use the USART peripheral through the STM32L1xx UART HAL and LL API, the LL API being used for performance improvement.	-	X	-	-
		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 STM32L1xx UART HAL and LL API, the LL API being used for performance improvement.	-	X	-	-
	Total number of examples_mix: 12			0	12	0	0
Applications	FatFs	FatFs_uSD	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.	-	-	-	X
	FreeRTOS	FreeRTOS_LowPower	How to enter and exit low-power mode with CMSIS RTOS API.	-	-	-	X
		FreeRTOS_Mail	How to use mail queues with CMSIS RTOS API.	-	-	-	X
		FreeRTOS_Mutexes	How to use mutexes with CMSIS RTOS API.	-	-	-	X
		FreeRTOS_Queues	How to use message queues with CMSIS RTOS API.	-	-	-	X
		FreeRTOS_Semaphore	How to use semaphores with CMSIS RTOS API.	-	-	-	X
		FreeRTOS_SemaphoreFromISR	How to use semaphore from ISR with CMSIS RTOS API.	-	-	-	X
		FreeRTOS_Signal	How to perform thread signaling using CMSIS RTOS API.	-	-	-	X
		FreeRTOS_SignalFromISR	How to perform thread signaling from an interrupt using CMSIS RTOS API.	-	-	-	X
		FreeRTOS_ThreadCreation	How to implement thread creation using CMSIS RTOS API.	X	X	X	X
		FreeRTOS_Timers	How to use timers of CMSIS RTOS API.	-	-	-	X
	STemWin	STemWin_HelloWorld	Simple "Hello World" example based on STemWin.	-	-	-	X

Level	Module name	Project name	Description	32L152CDISCOVERY	NUCLEO-L152RE	32L100CDISCOVERY	STM32L152D-EVAL
Applications	Touch Sensing	TouchSensing_Linear_hwacq	Use of the STMTouch driver with 1 linear sensor in hardware acquisition mode.	-	-	-	X
		TouchSensing_Linear_swacq	Use of the STMTouch driver with 1 linear sensor in software acquisition mode.	-	-	-	X
	USB_Device	CDC_Standalone	Use of the USB device application based on the Device Communication Class (CDC) and following the PSTN subprotocol. This application uses the USB Device and UART peripherals.	-	-	-	X
		CustomHID_Standalone	Use of the USB device application based on the Custom HID Class.	-	-	-	X
		DFU_Standalone	Compliant implementation of the Device Firmware Upgrade (DFU) capability to program the embedded Flash memory through the USB peripheral.	-	X	-	X
		HID_Standalone	Use of the USB device application based on the Human Interface (HID).	-	X	-	X
		MSC_Standalone	Use of the USB device application based on the mass storage class (MSC).	-	-	-	X
	Total number of applications: 24			1	3	1	19
Demonstrations	-	Adafruit_LCD_1_8_SD_Joystick	Demonstration firmware based on STM32Cube. This example helps you to discover STM32 Cortex-M devices that are plugged onto your STM32 Nucleo board.	-	X	-	-
		LED_Blinking	This demonstration uses USER button, LED3 and LED4 available on the board.	-	-	X	-
	Total number of demonstrations: 2			0	1	1	0
Total number of projects: 188				10	118	11	49

Revision history

Table 2. Document revision history

Date	Revision	Changes
27-May-2015	1	Initial release.
11-Jul-2016	2	Added SW4STM32 firmware in Section 2 STM32CubeL1 examples. Added support for Low-layer drivers.
26-Apr-2017	3	Updated Figure 1. STM32CubeL1 firmware components. Added: Templates_LL on all boards. HAL_TimeBase_TIM for STM32L152D_EVAL. Removed HAL_TimeBase for STM32L152D_EVAL.
13-Sep-2017	4	Updated Table 1. STM32CubeL1 firmware examples.
18-Jun-2019	5	Updated Table 1. STM32CubeL1 firmware examples.

Contents

List of tables