

Introduction to STM32Cube MCU Package examples for STM32H7Rx/7Sx MCUs

Introduction

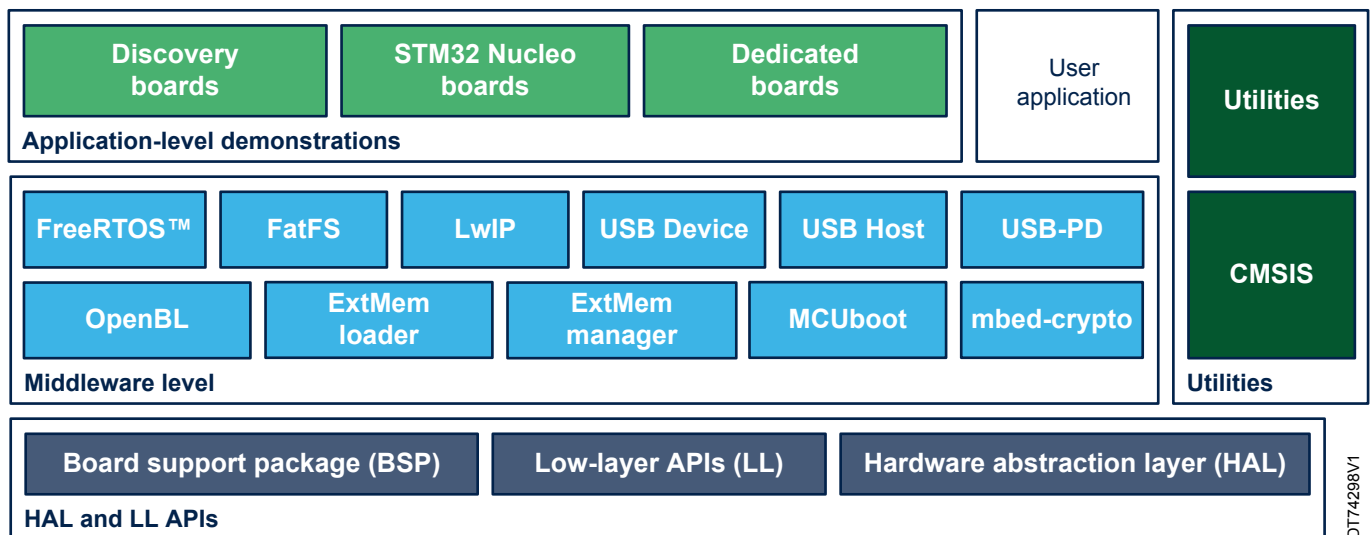
The **STM32CubeH7RS** MCU Package is delivered with a rich set of examples running on STMicroelectronics boards. The examples are organized by boards. They are provided with preconfigured projects for the main supported toolchains (refer to [Figure 1. STM32CubeH7RS firmware components](#)).

In the **STM32CubeH7RS** MCU Package, most examples and application projects are generated with the **STM32CubeMX** tool (starting from version v6.11.0) 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 their final application.

For more information about **STM32CubeMX**, refer to the *STM32CubeMX for STM32 configuration and initialization C code generation* user manual (UM1718).

Figure 1. STM32CubeH7RS firmware components



1 Reference documents

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

- The latest release of the [STM32CubeH7RS](#) MCU Package for the 32-bit microcontrollers in the STM32H7Rx/7Sx devices based on the Arm® Cortex®-M processor
- *Getting started with STM32CubeH7RS for STM32H7Rx/7Sx MCUs* (UM3294)
- *Description of STM32H7Rx/7Sx HAL and low-layer drivers* (UM3309)

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



2 STM32CubeH7RS examples

The examples are classified depending on the STM32Cube level that 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 or peripheral features and usage. They are organized per peripheral (one folder per peripheral, such as TIM). Their complexity level ranges from the basic usage of a given peripheral, such as PWM generation using a timer, to the integration of several peripherals, such as how to use DAC for a 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 are principally deployed on Nucleo boards.
- **Examples_MIX**
 These examples use only HAL, BSP, and LL drivers (middleware components are not used). They aim at demonstrating how to use both HAL and LL APIs in the same application in order to combine the advantages of both APIs:
 - HAL offers high-level function-oriented APIs with high portability level by hiding product/IPs complexity for end-users.
 - LL provides low-level APIs at the register level with better optimization.
 The examples are organized per peripheral (one folder for each peripheral, such as TIM) and are exclusively deployed on Nucleo boards.
- **Applications**
 The applications demonstrate product performance and how to use the available middleware stacks. They are organized by middleware (one folder per middleware, such as USB Host). The integration of applications that use several middleware stacks is also supported.
- **Template project**
 The template project is provided to enable the user to quickly build a firmware application using HAL and BSP drivers on a given board.
- **Template_LL project**
 The template LL projects are provided to enable the user to quickly build a firmware application using LL drivers on a given board.
- **Template_XIP project**
 The template XIP (execute in place) provides a reference template based on HAL drivers and external memory manager middleware that can be used to build any firmware application that can be downloaded in and executed from the external flash memory (subproject Appli). It boots from the internal flash memory and jumps to the application code in the external flash memory (subproject Boot).
- **Template_LRUN project**
 The template LRUN (load and run) provides a reference template based on HAL drivers and external memory manager middleware that can be used to build any firmware application that can be downloaded in the external flash memory (subproject Appli), and executed from the RAM memory after copy. It boots from the internal flash memory, copies the application from the external flash memory into the RAM, and jumps to the application code (subproject Boot).
- **Template_RoT project**
 The template RoT (root-of-trust) provides STiRoT and OEMiRoT boot path reference templates that can be used to build any firmware application with root-of-trust. Boot is performed through the selected STiRoT or OEMiRoT boot path after the authenticity and the integrity checks of the application.
- **Template_Board project**
 The template provides a reference template for the [NUCLEO-H7S3L8](#) board based on HAL and BSP drivers. These drivers can be used to build any firmware application that can be executed from the external flash memory (subproject Appli). It boots from the internal flash memory and jumps to the application code in the external flash memory (subproject Boot). This template was created from [STM32CubeMX](#) using the "Start My project from ST board" feature."

The examples are located under `STM32Cube_FW_H7RS_VX.Y.Z\Projects\.`

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

- *\`Boot`
*\`Inc` folder containing all the header files for the “boot” part
*\`Src` folder containing the code sources for the “boot” part
- *\`Appli`
*\`Inc` folder containing all the header files for the "user application" part
*\`Src` folder containing all the code sources for the "user application" part
*\`EWARM`, *\`MDK-ARM` and *\`STM32CubeIDE` folders, containing the preconfigured project for each toolchain.
*\`readme.html` and *\`README.md` describing the behavior of the example, and the environment required to run the example
*.ioc file that allows users to open firmware examples within [STM32CubeMX](#)
- \`ExtMemloader` folder containing files used to generate a binary able to download an application to an external memory.

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

The examples can be tailored to run on any compatible hardware: simply update the BSP drivers for your board, provided it has the same hardware functions (LED, LCD, pushbuttons, and others). The BSP is based on a modular architecture that can be easily ported to any hardware by implementing low-level routines.

[Table 1. STM32CubeH7RS firmware examples](#) contains the list of examples provided with the STM32CubeH7RS MCU Package.


In this table, the label  means that the projects are created using STM32CubeMX, the STM32Cube initialization code generator. Those projects can be opened with this tool to modify the projects themselves. The other projects are manually created to demonstrate the product features. Read the project *\`README.md` and *\`readme.html` files for user option bytes configuration.

Table 1. STM32CubeH7RS firmware examples

STM32CubeMX-generated examples are highlighted with the icon .

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Templates_Board	-	-	This project provides a reference template for the NUCLEO-H7S3L8 board. The template is based on the STM32Cube HAL API, and on the BSP drivers that can be used to build any firmware application to execute from external flash (subproject Appli). It boots from internal flash and jumps to the application code in external flash (subproject Boot).	-	-	
	Total number of templates_board: 1			0	0	1
Templates	-	Template	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application to execute from internal flash.		-	
		Template_LRUN	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application that executes an application stored on the NOR flash (subproject Appli). It boots from the internal flash, copies the application from the external NOR flash to the external PSRAM, and jumps to the application code in external PSRAM (subproject Boot).		-	
		Template_XIP	This project provides a reference template based on the STM32Cube HAL API that can be used to build any firmware application to execute from the external flash (subproject Appli). It boots from internal flash and jumps to the application code in external flash (subproject Boot).		-	
	Template_RoT	OEMiRoT_Appli	This project provides a OEMiRoT boot-path application example. Boot is performed through the OEMiRoT boot path after authenticity and integrity checks of the project firmware.		-	-
		STiRoT_Appli	This project provides a STiRoT boot-path reference template. Boot is performed through the STiRoT boot path after authenticity and the integrity checks of the project firmware and data images.		-	-
	Total number of templates: 8			5	0	3
Templates_LL	-	Starter project	This project provides a reference template through the LL API that can be used to build any firmware application.		-	
	Total number of templates_LL: 2			1	0	1
Examples	-	BSP	This example provides a description on how to use the different BSP drivers.		-	-
	ADC	ADC_AnalogWatchdog	This example describes 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.	-	-	
		ADC_DifferentialMode	This example describes how to configure and use the ADC1 to convert an external analog input in differential mode. Difference between external voltage on VinN and VinP.	-	-	
		ADC_DiscontinuousConversion_TriggerSW	This example describes how to use an ADC peripheral to perform multiple conversions from a different ADC channel, one at a time after each software trigger.	-	-	
		ADC_SingleConversion_TriggerSW	This example describes how to use the ADC to convert a single channel at each software start. The conversion is performed using the programming model: polling.	-	-	



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	ADC	ADC_SingleConversion_TriggerSW_HPDMa	This example describes how to use an ADC peripheral to perform a single ADC conversion on a channel at each software start. HPDMA transfers the converted data into a table in the SRAM memory.	-	-	MX
		ADC_SingleConversion_TriggerSW_IT	How to use ADC to convert a single channel at each software start. Conversion is performed using a programming model: interrupt.	-	-	MX
	CORDIC	CORDIC_Exp_DMA	How to use the CORDIC peripheral to calculate the exponential of a value.	-	-	MX
		CORDIC_Ln_DMA	How to use the CORDIC peripheral to calculate the natural logarithm of a value.	-	-	MX
		CORDIC_Phase_DMA	How to use the CORDIC peripheral to calculate an array of phases in DMA mode.	-	-	MX
		CORDIC_Sin_DMA	How to use the CORDIC peripheral to calculate an array of sines in DMA mode.	-	-	MX
		CORDIC_Sqrt_DMA	How to use the CORDIC peripheral to calculate an array of SQRT in DMA mode.	-	-	MX
	Cortex	CORTEXM_CACHE	This project provides a Cortex-M cache example based on the CMSIS API that can be used to build any firmware application to execute from internal flash.	-	-	MX
		CORTEXM_MPU	Presentation of the MPU features. This example configures the MPU attributes of different MPU regions. Then it configures a memory area as privileged read-only, and attempts to perform read and write operations in different modes.	-	-	MX
		CORTEXM_ModePrivilege	How to modify the thread-mode privilege access and stack. Thread mode is entered on reset or when returning from an exception.	-	-	MX
		CORTEXM_SysTick	How to use the default SysTick configuration with a 1 ms timebase to toggle LEDs.	-	-	MX
	CRC	CRC_Bytes_Stream_7bit_CRC	How to configure the CRC using the HAL API. The CRC (cyclic redundancy check) calculation unit computes 7-bit CRC codes derived from buffers of 8-bit data (bytes). The user-defined generating polynomial is manually set to 0x65, that is, $X^7 + X^6 + X^5 + X^2 + 1$, as used in the train communication network, IEC 60870-5[17].	-	-	MX
		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 (0x4C1 1DB7).	-	-	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
	CRYP	CRYP_SAES_ECB_CBC	How to use the secure AES coprocessor (SAES) peripheral to encrypt and decrypt data using AES ECB and CBC algorithms.	MX	-	-
		CRYP_SAES_ECB_CBC_AHK	How to use the Secure AES co-processor (SAES) peripheral to encrypt and decrypt data using AES ECB and CBC algorithms with AHK (application hardware keys) Key.	-	-	MX

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	CRYPT	CRYPT_SAES_GCM	This project describes how to use the CRYPT peripheral to encrypt and decrypt data using AES with Galois/counter mode (GCM). It boots from internal flash and jumps to the application code in external flash (subproject Boot).	-	-	MX
		CRYPT_SAES_SharedKey	How to use the secure AES coprocessor (SAES) peripheral to share application keys with the AES peripheral.	MX	-	-
		CRYPT_SAES_WrapKey	How to use the secure AES coprocessor (SAES) peripheral to wrap application keys using the hardware secret key DHUK, then use it to encrypt in polling mode.	MX	-	-
	DCMIPP	DCMIPP_ContinuousDBM	This example shows how to use the DCMIPP peripheral in continuous double buffering mode and it is based on the STM32Cube HAL API.	MX	-	-
		DCMIPP_ContinuousMode	This example shows how to use the DCMIPP peripheral in continuous mode and it is based on the STM32Cube HAL API.	MX	-	-
		DCMIPP_JPEGSnapshotMode	This example shows how to use the DCMIPP peripheral in snapshot and JPEG mode, and it is based on the STM32Cube HAL API.	MX	-	-
	DMA	DMA_DataHandling	This example describes how to use the DMA controller to do data handling between transferred data from the source and transferred to the destination through the HAL API.	-	-	MX
		DMA_FLASHToRAM	This example describes how to use DMA to transfer a word data buffer from flash memory to embedded SRAM through the HAL API.	-	-	MX
		DMA_LinkedList	This example describes how to use the DMA to perform a list of transfers. The transfer list is organized as a linked-list. Each time the current transfer ends the DMA automatically reloads the next transfer parameters, and starts it (without CPU intervention).	-	-	MX
		DMA_RepeatedBlock	How to configure and use the DMA HAL API to perform repeated block transactions.	-	-	MX
		DMA_Trigger	How to configure and use the DMA HAL API to perform DMA triggered transactions.	-	-	MX
	DMA2D	DMA2D_BlendingWithAlphaInversion	This example provides a description of how to configure a DMA2D peripheral in memory-to-memory with blending transfer and alpha inversion mode. It boots from internal flash and jumps to the application code in external flash.	MX	-	-
		DMA2D_MemToMemWithBlending	This example provides a description of how to configure a DMA2D peripheral in memory-to-memory with blending transfer mode.	MX	-	-
		DMA2D_MemToMemWithBlendingAndCLUT	This example provides a description of how to configure a DMA2D peripheral in memory-to-memory blending transfer mode with indexed 256 color images (L8).	MX	-	-
		DMA2D_MemoryToMemory	This example provides a description of how to configure the DMA2D peripheral in memory-to-memory transfer mode.	MX	-	-
		DMA2D_RegToMemWithLCD	This example provides a description of how to configure the DMA2D peripheral in register-to-memory transfer mode and display the result on LCD.	MX	-	-



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	DTS	DTS_GetTemperature	How to configure and use the DTS to get the temperature of the die.	-	-	MX
	FDCAN	FDCAN_Adaptive_Bitrate_Receiver	This example describes how to configure the FDCAN peripheral to adapt to different CAN bit rates using restricted mode.	-	-	MX
		FDCAN_Adaptive_Bitrate_Transmitter	This example describes how to configure the FDCAN peripheral to adapt to different CAN bit rates using restricted mode.	-	-	MX
		FDCAN_Classic_Frame_Networking	This example describes how to configure the FDCAN peripheral to send and receive classic CAN frames between two FDCAN units.	-	-	MX
		FDCAN_Com_IT	This example describes how to configure the FDCAN peripheral to achieve interrupt process communication between two FDCAN units.	-	-	MX
		FDCAN_Com_Polling	This example describes how to configure the FDCAN peripheral to achieve polling process communication between two FDCAN units.	-	-	MX
		FDCAN_Loopback	This example describes how to configure the FDCAN peripheral to operate in loopback mode.	-	-	MX
		FDCAN_Power_Down	This example describes the functionality of the power down mode in the FDCAN peripheral.	-	-	MX
	FLASH	FLASH_ECC_Error_Generation	How to use the flash HAL API to manage ECC errors.	MX	-	MX
		FLASH_EraseProgram	How to configure and use the flash HAL API to erase and program the internal flash memory. 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.	-	-	MX
		FLASH_OBK_Program	How to use the flash HAL API to program option bytes keys. After reset, the flash memory controller and option byte access are locked. Dedicated functions are used to enable the flash memory controller's register and option bytes access.	-	-	MX
	FMC	FMC_NOR	This project is targeted to run on a STM32H7S7xx device on the STM32H7RS_CUSTOM_HW board from STMicroelectronics.	-	MX	-
		FMC_SDRAM	This project is targeted to run on a STM32H7S7xx device on the STM32H7RS_CUSTOM_HW board from STMicroelectronics.	-	MX	-
		FMC_SRAM	This project is targeted to run on a STM32H7S7xx device on the STM32H7RS_CUSTOM_HW board from STMicroelectronics.	-	MX	-
	GFXTIM	GFXTIM_AbsoluteTimer_FrameCount	This example demonstrates the use of the GFXTIM peripheral to generate interrupts for specific absolute frames. It boots from internal flash and jumps to the application code in external flash.	MX	-	-
		GFXTIM_AbsoluteTimer_LineCount	This example demonstrates the use of the GFXTIM peripheral to generate interrupts for specific absolute lines. It boots from internal flash and jumps to the application code in external flash.	MX	-	-

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	GFXTIM	GFXTIM_EventGenerator_DisplayUpdate	This example demonstrates the use of the GFXTIM peripheral to generate a complex event and update the LTDC buffer accordingly using the DMA2D. It boots from internal flash and jumps to the application code in external flash.	MX	-	-
		GFXTIM_RelativeTimer_FrameCount	This example demonstrates the use of the GFXTIM peripheral to generate interrupts for specific relative frames. It boots from internal flash and jumps to the application code in external flash.	MX	-	-
		GFXTIM_Watchdog_Alarm_PreAlarm	This example demonstrates the use of the GFXTIM peripheral to generate interrupts on alarm and prealarm events. It boots from internal flash and jumps to the application code in external flash.	MX	-	-
	GPIO	GPIO_EXTI	How to configure external interrupt lines.	-	-	MX
		GPIO_IOToggle	How to configure and use GPIOs through the HAL API.	MX	-	MX
	HAL	HAL_TimeBase_RTC_WKUP	How to customize HAL using RTC wakeup as main source of time base, instead of SysTick.	-	-	MX
		HAL_TimeBase_TIM	How to customize HAL using a general-purpose timer as the main source of time base instead of SysTick.	-	-	MX
	HASH	HASH_SHA256	This project describes how to use the HASH peripheral to hash data using the SHA-256 algorithm.	-	-	MX
		HASH_SHA384	This project describes how to use the HASH peripheral to hash data using the SHA-384 algorithm.	-	-	MX
		HASH_SHA512	This project describes how to use the HASH peripheral to hash data using the SHA-512 algorithm.	-	-	MX
	I2C	I2C_TwoBoards_AdvComIT	How to handle several I ² C data buffer transmission/reception between a controller and a target device, using an interrupt.	-	-	MX
		I2C_TwoBoards_ComDMA	How to handle I ² C data buffer transmission/reception between two boards, via the DMA.	-	-	MX
		I2C_TwoBoards_ComIT	How to handle I ² C data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
		I2C_TwoBoards_ComPolling	How to handle I ² C data buffer transmission/reception between two boards, in polling mode.	-	-	MX
		I2C_TwoBoards_RestartAdvComIT	How to perform multiple I ² C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	-	-	MX
		I2C_TwoBoards_RestartComIT	How to handle single I ² C data buffer transmission/reception between two boards, in interrupt mode and with restart condition.	-	-	MX



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	I2C	I2C_WakeUpFromStop	How to handle I2C data buffer transmission/reception between two boards, using an interrupt when the device is in Stop mode.	-	-	MX
	I3C	I3C_Controller_Direct_Command_DMA	How to handle a direct command procedure between an I3C controller and an I3C target, using DMA.	-	-	MX
		I3C_Controller_ENTDAA_IT	How to handle an ENTDAAs procedure between an I3C controller and one or more I3C targets.	-	-	MX
		I3C_Controller_HotJoin_IT	How to handle a HOTJOIN procedure between an I3C controller and I3C targets.	-	-	MX
		I3C_Controller_I2C_ComDMA	How to handle I ² C communication as I3C controller data buffer transmission/reception between two boards, using the DMA.	-	-	MX
		I3C_Controller_IBI_Wakeup_IT	How to handle an in-band-interrupt event between an I3C controller in low power mode and I3C targets.	-	-	MX
		I3C_Controller_InBandInterrupt_IT	How to handle an in-band-interrupt event between an I3C controller and I3C targets.	-	-	MX
		I3C_Controller_Private_Command_IT	How to handle I3C as controller data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
		I3C_Controller_Switch_To_Target	How to handle a controller role request direct command procedure between an I3C controller and an I3C target, using an interrupt.	-	-	MX
		I3C_Sensor_Direct_Command_DMA	How to handle a direct command procedure between NUCLEO-H7S3L8 and a X-NUCLEO-IKS01A3 , using the DMA.	-	-	MX
		I3C_Sensor_Private_Command_IT	How to handle I3C as controller data buffer transmission/reception between NUCLEO-H7S3L8 and a X-NUCLEO-IKS01A3, using an interrupt.	-	-	MX
		I3C_Target_Direct_Command_DMA	How to handle a direct command procedure between an I3C controller and an I3C target, using a controller in the DMA.	-	-	MX
		I3C_Target_ENTDAA_IT	How to handle an ENTDAAs procedure between an I3C controller and one or more I3C targets.	-	-	MX
		I3C_Target_HotJoin_IT	How to handle a HOTJOIN procedure to an I3C controller.	-	-	MX
		I3C_Target_I2C_ComDMA	How to handle I ² C data buffer transmission/reception between two boards, using the DMA.	-	-	MX
		I3C_Target_IBI_Wakeup_IT	How to handle an in-band-interrupt procedure to an I3C controller in stop mode.	-	-	MX



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	I3C	I3C_Target_InBandInterrupt_IT	How to handle an in-band-interrupt event to an I3C controller.	-	-	MX
		I3C_Target_Private_Command_IT	How to handle I3C as target data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
		I3C_Target_Switch_To_Controller	How to handle a controller-role request procedure to an I3C controller.	-	-	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 lapse of time.	-	-	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 lapse of time.	-	-	MX
	JPEG	JPEG_DecodingFromXSPI_DMA	This example demonstrates how to decode a JPEG image stored in the external flash (XSPI2) using the JPEG hardware decoder in DMA mode. The decoded image is stored in the internal SRAM.	MX	-	-
		JPEG_EncodingFromXSPI_DMA	This example demonstrates how to encode an RGB image stored in the external flash (XSPI2). It is done using the JPEG hardware encoder in DMA mode, and saves it in the internal flash.	MX	-	-
	LPTIM	LPTIM_PWM_LSE	How to configure and use the LPTIM peripheral through the HAL LPTIM API. This configuration is done by using LSE as counter clock, to generate a PWM signal, and in a low-power mode.	-	-	MX
	LTDC	LTDC_ColorKeying	This example describe how to enable and use the LTDC color keying functionality.	MX	-	-
		LTDC_Display_2Layers	This example describes how to configure the LTDC peripheral to display two layers at the same time.	MX	-	-
	MCE	MCE_AESEncryptDecryptData	This project provides a description of how to encrypt and decrypt data from external memory (PSRAM).	MX	-	-
		MCE_ExecuteAESCryptedCode	This project provides a description of how to run encrypted application code from external flash. The decryption is performed on-the-fly using the MCE peripheral (AES encryption algorithm).	MX	-	-
		MCE_ExecuteNoekeonCryptedCode	This project provides a description of how to run encrypted application code from external flash (0x7000 0000-0x7FFF FFFF). The decryption is performed on-the-fly using the MCE peripheral (Noekeon encryption algorithm).	MX	-	-
		MCE_NoekeonEncryptDecryptData	This project provides a description of how to encrypt and decrypt data from external memory (NOR).	MX	-	-
	MDF	ADF_AudioSoundDetector	How to use the MDF HAL API (ADF instance) to use audio sound activity detection.	MX	-	-
PKA	PKA_ECCDoubleBaseLadder	How to use the PKA to run ECC double base ladder operation.	-	-	MX	

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	PKA	PKA_ECCProjective2Affine	How to use the PKA to run ECC projections to affine operation. This project runs from the external flash memory. It is launched from a first boot stage and inherits from this boot project the following: configuration caches, MPU regions [region 0 and 1], system clock at 600 MHz, and external memory interface at the highest speed.	-	-	MX
		PKA_ECDSA_Sign	How to compute a signed message regarding the elliptic curve digital signature algorithm (ECDSA).	-	-	MX
		PKA_ModExpProtected_IT	How to use the PKA to run a protected modular exponentiation operation. This project is targeted to run on STM32H7S3L8Hx devices on NUCLEO-H7S3L8 board from STMicroelectronics.	-	-	MX
	PSSI	PSSI_Master_Com	How to handle a communication procedure using two boards with PSSI.	-	-	MX
		PSSI_Master_ComDMA	How to handle a DMA communication procedure using two boards with PSSI.	-	-	MX
		PSSI_Master_Single_Com	How to handle a single communication procedure using two boards with PSSI in polling mode.	-	-	MX
		PSSI_Master_Single_ComDMA	How to handle a single DMA communication procedure using two boards with PSSI.	-	-	MX
		PSSI_Slave_Com	How to handle a communication procedure using two boards with PSSI.	-	-	MX
		PSSI_Slave_ComDMA	How to handle a DMA communication procedure using two boards with PSSI.	-	-	MX
		PSSI_Slave_Single_Com	How to handle a single communication procedure using two boards with PSSI in polling mode.	-	-	MX
		PSSI_Slave_Single_ComDMA	How to handle a single DMA communication procedure using two boards with PSSI.	-	-	MX
		PSSI_Transmit	How to handle a simple transmit procedure with PSSI.	-	-	MX
		PSSI_Transmit_DMA	How to handle a DMA transmit procedure with PSSI.	-	-	MX
	PWR	PWR_SLEEP	How to enter the Sleep mode and wake up from this mode by using an interrupt.	-	-	MX
		PWR_STANDBY	How to enter the Standby mode and wake up from this mode by using 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 wake-up timer.	-	-	MX

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	PWR	PWR_STOP	How to enter the Stop mode and wake-up from this mode by using an interrupt.	-	-	MX
		PWR_STOP_RTC	How to enter the Stop mode and wake-up from this mode by using an external reset or the RTC wake-up timer.	-	-	MX
	RAMECC	RAMECC_ErrorCount	How to configure and use the RAMECC HAL API to manage ECC errors via the RAMECC peripheral.	-	-	MX
	RCC	RCC_CRs_Synchronization_IT	Configuration of the clock recovery system (CRS) in interrupt mode, using the RCC HAL API.	-	-	MX
		RCC_CRs_Synchronization_Polling	Configuration of the clock recovery system (CRS) in polling mode, using the RCC HAL API.	-	-	MX
		RCC_ClockConfig	Configuration of the system clock (SYSCLK) and modification of the clock settings in run mode, using the RCC HAL API.	MX	-	MX
		RCC_LSEConfig	Enabling/disabling of the low-speed external(LSE) RC oscillator (about 32 kHz) at runtime, using the RCC HAL API.	-	-	MX
		RCC_LSIConfig	How to enable/disable the low-speed internal (LSI) RC oscillator (about 32 kHz) at runtime, 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
		RNG_MultiRNG_IT	Configuration of the RNG using the HAL API. This example uses RNG interrupts to generate 32-bit long random numbers.	-	-	MX
	RTC	RTC_ActiveTamper	Configuration of the active tamper detection with backup registers erase.	-	-	MX
		RTC_Alarm	Configuration and generation of an RTC alarm using the RTC HAL API.	-	-	MX
		RTC_Calendar	Configuration of the calendar using the RTC HAL API.	MX	-	MX
		RTC_LSI	Use of the LSI clock source autocalibration to get a precise RTC clock.	-	-	MX
		RTC_LowPower_STANDBY_WUT	How to periodically enter and wake up from Standby mode thanks to the RTC wake-up timer (WUT).	-	-	MX
		RTC_Tamper	Configuration of the tamper detection with backup registers erase.	-	-	MX

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	RTC	RTC_TimeStamp	Configuration of the RTC HAL API to demonstrate the timestamp feature.	-	-	MX
	SD	SD_ReadWrite_DMA	This example performs some write and read transfers to the SD card with SDMMC peripheral internal DMA mode.	MX	-	-
		SD_ReadWrite_DMALinkedList	This example performs some write and read transfers to SD Card with SDMMC IP internal DMA mode based on Linked list feature.	MX	-	-
	SMBUS	SMBUS_TwoBoards_ComIT_Master	How to handle SMBUS data buffer transmission/reception between two boards, in interrupt mode.	-	-	MX
		SMBUS_TwoBoards_ComIT_Slave	How to handle SMBUS data buffer transmission/reception between two boards, in interrupt mode.	-	-	MX
	SPDIFRX	SPDIFRX_Loopback	How to configure and use the SPDIFRX peripheral to receive a data flow.	-	-	MX
	SPI	SPI_ACCEL_ComPolling	This example describes how to perform SPI transmission/reception in polling mode to program an accelerometer. This example can use different SPI accelerometers from STMicroelectronics.	-	-	MX
		SPI_EEPROM_ComPolling	This example describes how to perform SPI data buffer transmission/reception in polling mode. The communication uses an SPI EEPROM memory.	-	-	MX
		SPI_FullDuplex_ComDMA_Master	Data buffer transmission/reception between two boards via SPI using DMA.	-	-	MX
		SPI_FullDuplex_ComDMA_Slave	Data buffer transmission/reception between two boards via SPI using DMA.	-	-	MX
		SPI_FullDuplex_ComIT_Master	Data buffer transmission/reception between two boards via SPI using interrupt mode.	-	-	MX
		SPI_FullDuplex_ComIT_Slave	Data buffer transmission/reception between two boards via SPI using interrupt mode.	-	-	MX
		SPI_FullDuplex_ComPolling_Master	Data buffer transmission/reception between two boards via SPI using polling mode.	-	-	MX
		SPI_FullDuplex_ComPolling_Slave	Data buffer transmission/reception between two boards via SPI using polling mode.	-	-	MX
	TIM	TIM_InputCapture_DMA	This example demonstrates how to measure the LSI clock frequency thanks to the DMA interface of the TIM15 timer instance.	-	-	MX
		TIM_OCActive	Configuration of the TIM peripheral in output-compare active mode. When the counter matches the capture/compare register, the corresponding output pin is set to its active state.	-	-	MX

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	TIM	TIM_OCInactive	Configuration of the TIM peripheral in output compare inactive mode with the corresponding interrupt requests for each channel.	-	-	MX
		TIM_OCToggle	Configuration of the TIM peripheral to generate four different signals at four different frequencies.	-	-	MX
		TIM_PWMInput	How to use the TIM peripheral to measure the frequency and duty cycle of an external signal.	-	-	MX
		TIM_PWMOutput	This example shows how to configure the TIM peripheral in PWM (pulse width modulation) mode.	-	-	MX
	UART	UART_Printf	Rerouting of the C library print function to the UART.	-	-	MX
		UART_ReceptionToIdle_CircularDMA	How to use the HAL UART API for reception to IDLE event in circular DMA mode.	-	-	MX
		UART_TwoBoards_ComDMA	UART transmission (transmit/receive) in DMA mode between two boards.	-	-	MX
		UART_TwoBoards_ComDMAlinkedlist	UART transmission (transmit/receive) in DMA mode using linked list between two boards.	-	-	MX
		UART_TwoBoards_ComIT	UART transmission (transmit/receive) in interrupt mode between two boards.	-	-	MX
		UART_TwoBoards_ComPolling	UART transmission (transmit/receive) in polling mode between two boards.	-	-	MX
		UART_WakeUpFromStopUsingFIFO	Configuration of an UART to wake up the MCU from Stop mode with a FIFO level when a given stimulus is received.	-	-	MX
	USART	USART_SlaveMode	This example describes a USART-SPI communication (transmit/receive) between two boards where the USART is configured as a target.	-	-	MX
		USART_SlaveMode_DMA	This example describes a USART-SPI communication (transmit/receive) with DMA between two boards where the USART is configured as a target.	-	-	MX
	WWDG	WWDG_Example	Configuration of the HAL API to periodically update the WWDG counter and simulate a software fault. This fault generates an MCU WWDG reset when a predefined time period has elapsed.	-	-	MX
	XSPI	XSPIM_SwappedMode	This project provides a description of how to configure the XSPIM I/O manager peripheral and communicate with external memories in swapped mode.	MX	-	-
		XSPI_NOR_AutoPolling_DTR	How to use an XSPI NOR memory in automatic polling mode.	MX	-	-

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples	XSPI	XSPI_NOR_ReadWhileWrite_DTR	How to write and read data using XSPI NOR memory.	MX	-	-
		XSPI_PSRAM_MemoryMapped	How to use an XSPI PSRAM in memory-mapped mode.	MX	-	-
	Total number of examples: 171			36	3	132
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
		ADC_Oversampling_Init	How to use an ADC peripheral with oversampling.	-	-	MX
		ADC_SingleConversion_TriggerSW_IT_Init	How to use ADC to convert a single channel at each software start. The conversion is performed using a programming model: interrupt.	-	-	MX
		ADC_SingleConversion_TriggerSW_Init	How to use ADC to convert a single channel at each software start. The conversion is performed using a programming model: polling.	-	-	MX
	CORDIC	CORDIC_CosSin	How to use the CORDIC peripheral to calculate cosine and sine.	-	-	MX
	CORTEX	CORTEX_MPU	Presentation of the MPU features. This example configures the MPU attributes of different MPU regions. Then it configures a memory area as privileged read-only, and attempts to perform read and write operations in different modes.	-	-	MX
	CRC	CRC_CalculateAndCheck	How to configure the CRC calculation unit to compute a CRC code for a given data buffer, based on a fixed generator polynomial (default value 0x4C1 1DB7). The peripheral initialization is done using LL unitary service functions for optimization purposes (performance and size).	-	-	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
	CRS	CRS_Synchronization_IT	How to configure the clock recovery system in IT mode through the STM32H7Rx/7Sx devices CRS LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		CRS_Synchronization_Polling	How to configure the clock recovery system in polling mode through the STM32H7Rx/7Sx devices CRS LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	DMA	DMA_CopyFromFlashToMemory_Init	How to use a DMA channel to transfer a word data buffer from flash memory to an embedded SRAM. The peripheral initialization uses LL initialization functions to demonstrate LL init usage.	-	-	MX
	DMA2D	DMA2D_MemoryToMemory	This example provides a description of how to configure a DMA2D peripheral in memory-to-memory transfer mode.	-	-	MX
	EXTI	EXTI_ToggleLedOnIT_Init	This example describes how to configure the EXTI and use GPIOs to toggle the user LEDs available on the board when a user button is pressed. This example is based on the STM32H7Rx/7Sx devices LL API. Peripheral initialization is done using the LL initialization function to demonstrate LL init usage.	-	-	MX



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8	
Examples_LL	GPIO	GPIO_InfiniteLedToggling_Init	How to configure and use GPIOs to toggle the on-board user LEDs every 250 ms. This example is based on the STM32H7Rx/7Sx devices LL API. The peripheral is initialized with an LL initialization function to demonstrate LL init usage.	-	-	MX	
	I2C	I2C_OneBoard_Communication_IT_Init	How to handle the reception of one data byte from an I2C target device by a controller device. Both devices operate in interrupt mode.	-	-	MX	
	I3C	I3C_Controller_Direct_Command_IT	I3C_Controller_Direct_Command_IT	How to handle a direct command procedure between an I3C controller and an I3C target, using IT.	-	-	MX
		I3C_Controller_Direct_Command_Polling	I3C_Controller_Direct_Command_Polling	How to handle a direct command procedure between an I3C controller and an I3C target, using polling.	-	-	MX
		I3C_Controller_Private_Command_IT	I3C_Controller_Private_Command_IT	How to handle I3C as controller data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
		I3C_Controller_WakeUpFromStop	I3C_Controller_WakeUpFromStop	How to handle I3C as controller data buffer transmission/reception between a target in stop mode, using interrupt.	-	-	MX
		I3C_Target_Direct_Command_IT	I3C_Target_Direct_Command_IT	How to handle a direct command procedure between an I3C controller and an I3C target, using the controller in interrupt.	-	-	MX
		I3C_Target_Direct_Command_Polling	I3C_Target_Direct_Command_Polling	How to handle a direct command procedure between an I3C controller and an I3C target, using the controller in polling.	-	-	MX
		I3C_Target_Private_Command_IT	I3C_Target_Private_Command_IT	How to handle I3C as target data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
		I3C_Target_WakeUpFromStop	I3C_Target_WakeUpFromStop	How to handle I3C as target data buffer transmission/reception between two boards, using an interrupt.	-	-	MX
	IWDG	IWDG_RefreshUntilUserEvent_Init	How to configure the IWDG peripheral to ensure periodical counter update and generate an MCU IWDG reset when a USER push-button is pressed. The peripheral is initialized with LL unitary service functions to optimize for performance and size.	-	-	MX	
	LPTIM	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 STM32H7Rx/7Sx devices LPTIM LL API. The peripheral is initialized with an LL initialization function to demonstrate LL init usage.	-	-	MX	
	LPUART	LPUART_WakeUpFromStop_Init	Configuration of GPIO and LPUART peripherals to allow characters received on the 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 the LL initialization function to demonstrate LL init usage.	-	-	MX	
	PWR	PWR_EnterStandbyMode	PWR_EnterStandbyMode	How to enter the Standby mode and wake up from this mode by using an external reset or a wakeup pin.	-	-	MX
		PWR_EnterStopMode	PWR_EnterStopMode	How to enter the Stop mode.	-	-	MX



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples_LL	RCC	RCC_OutputSystemClockOnMCO	Configuration of the MCO pin (PA8) to output the system clock.	-	-	MX
		RCC_UseHSI_PLLasSystemClock	Modification of the PLL parameters in runtime.	-	-	MX
	RTC	RTC_Alarm_Init	Configuration of the RTC LL API to configure and generate an alarm using the RTC peripheral. The peripheral initialization uses the LL initialization function.	-	-	MX
		RTC_Calendar_Init	Configuration of the LL API to set the RTC calendar. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		RTC_ExitStandbyWithWakeUpTimer_Init	How to periodically enter and wake up from Standby mode thanks to the RTC wake-up timer (WUT).	-	-	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
		RTC_TimeStamp_Init	Configuration of the timestamp using the RTC LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	SPI	SPI_OneBoard_HalfDuplex_IT_Init	Configuration of GPIO and SPI peripherals to transmit bytes from an SPI controller device to an SPI target device in interrupt mode. This example is based on the STM32H7Rx/7Sx devices SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		SPI_TwoBoards_FullDuplex_IT_Master_Init	Data buffer transmission and reception via SPI using interrupt mode. This example is based on the STM32H7Rx/7Sx devices SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		SPI_TwoBoards_FullDuplex_IT_Slave_Init	Data buffer transmission and reception via SPI using interrupt mode. This example is based on the STM32H7Rx/7Sx devices SPI LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	TIM	TIM_BreakAndDeadtime_Init	Configuration of the TIM peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined deadtime value, use the break feature, and lock the break and dead-time configuration.	-	-	MX
		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 STM32H7Sxx TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		TIM_InputCapture_Init	Use of the TIM peripheral to measure a periodic signal frequency provided either by an external signal generator or by another timer instance. This example is based on the STM32H7Rx/7Sx devices TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		TIM_OnePulse_Init	Configuration of a timer to generate a positive pulse in Output Compare mode with a length of tPULSE and after a delay of tDELAY.	-	-	MX

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples_LL	TIM	TIM_OutputCompare_Init	Configuration of the TIM peripheral to generate an output waveform in different output compare modes. This example is based on the STM32H7Rx/7Sx devices TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
		TIM_PWMOutput_Init	Use of a timer peripheral to generate a PWM output signal and update the PWM duty cycle. This example is based on the STM32H7Rx/7Sx devices TIM LL API. The peripheral initialization uses the LL initialization function to demonstrate LL Init.	-	-	MX
		TIM_TimeBase_Init	Configuration of the TIM peripheral to generate a timebase. This example is based on the STM32H7Rx/7Sx devices TIM LL API. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).	-	-	MX
	USART	USART_Com_Rx_IT_Continuous_Init	This example shows how to configure a GPIO and a 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 purposes (performance and size).	-	-	MX
		USART_Com_Rx_IT_Continuous_VCP_Init	This example shows how to configure a GPIO and USART peripheral for continuously receiving characters from a HyperTerminal (PC) in asynchronous mode using interrupt mode. Peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	-	MX
		USART_Com_Rx_IT_Init	This example shows how to configure a GPIO and USART peripheral for receiving characters from HyperTerminal (PC) in asynchronous mode using interrupt mode. Peripheral initialization is done using the LL initialization function to demonstrate LL init usage.	-	-	MX
		USART_Com_Tx_IT_Init	This example shows how to configure a GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in interrupt mode. This example is based on STM32H7Rx/7Sx devices USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	-	MX
		USART_Com_Tx_IT_VCP_Init	This example shows how to configure a GPIO and USART peripheral to send characters asynchronously to HyperTerminal (PC) in interrupt mode. This example is based on STM32H7Rx/7Sx devices USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	-	MX
		USART_Com_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 STM32H7Rx/7Sx devices USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	-	MX
		USART_Com_Tx_VCP_Init	This example shows how to configure GPIO and USART peripherals to send characters asynchronously to an HyperTerminal (PC) in polling mode. If the transfer could not be completed within the allocated time, a timeout allows to exit from the sequence with a timeout error code. This example is based on STM32H7Rx/7Sx devices USART LL API. Peripheral initialization is done using LL unitary services functions for optimization purposes (performance and size).	-	-	MX
UTILS	UTILS_ConfigureSystemClock	Use of UTILS LL API to configure the system clock using PLL with HSI as source clock.	-	-	MX	




Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Examples_LL	UTILS	UTILS_ReadDeviceInfo	This example reads the UID, device ID and revision ID, and saves them into a global information buffer.	-	-	
	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).	-	-	
	Total number of examples_LL: 55			0	0	55
Examples_MIX	ADC	ADC_SingleConversion_TriggerSW_IT	How to use ADC to convert a single channel at each software start, conversion performed using a programming model interrupt.	-	-	
	DMA	DMA_FLASHToRAM	How to use a DMA to transfer a word data buffer from flash memory to embedded SRAM through the STM32H7Rx/7Sx devices DMA HAL and LL API. The LL API is used for performance improvement.	-	-	
	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 STM32H7RSxx RTC and RCC HAL, and LL API (LL API use for maximizing performance).	-	-	
		PWR_STOP	How to enter the stop mode and wake up from this mode by using an external reset or wake-up interrupt; All the RCC function calls use the RCC LL API for minimizing footprint and maximizing performance.	-	-	
	SPI	SPI_FullDuplex_Compolling_Master	Data buffer transmission/reception between two boards via SPI using polling mode.	-	-	
		SPI_FullDuplex_Compolling_Slave	Data buffer transmission/reception between two boards via SPI using polling mode.	-	-	
	TIM	TIM_PWMInput	Use of the TIM peripheral to measure an external signal frequency and duty cycle.	-	-	
	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 STM32H7Rx/7Sx devices UART HAL and LL API, the LL API being used for performance improvement.	-	-	
		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 STM32H7Rx/7Sx devices UART HAL and LL API, the LL API being used for performance improvement.	-	-	
	Total number of examples_mix: 9			0	0	9
Applications	-	OpenBootloader	This application exploits the OpenBootloader middleware to demonstrate how to develop an IAP application, and how to use it.		-	-
	FatFs	FatFs_MultiAccess	This application provides a description on how to use STM32Cube firmware with a FatFs middleware component as a generic FAT file system module. To develop an application exploiting FatFs offered features with a microSD™ drive in RTOS mode configuration.		-	-
		FatFs_uSD_RTOS	This application provides a description on how to use STM32Cube firmware with a FatFs middleware component as a generic FAT file system module. To develop an application exploiting FatFs offered features with a microSD™ drive in RTOS mode configuration.		-	-



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Applications	FatFs	FatFs_uSD_Standalone	How to use STM32Cube firmware with a 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	-	-
	FreeRTOS™	FreeRTOS_MPU	This application demonstrates the use of the MPU with FreeRTOS™ to control memory/peripheral access for tasks.	-	-	X
		FreeRTOS_Mutex	This application demonstrates the use of mutexes to serialize access to a shared resource.	-	-	MX
		FreeRTOS_Queue_ThreadFlags	This application demonstrates the use of message queues, and thread flags with CMSIS_RTOS2 API.	-	-	MX
		FreeRTOS_Semaphore_LowPower	This application demonstrates the use of FreeRTOS™ tickless low power mode and semaphores.	-	-	MX
		LwIP	LwIP_HTTP_Server_Netconn_RTOS	This application guides STM32Cube HAL API users to run an http server application based on the Netconn API of the LwIP TCP/IP stack.	-	-
	LwIP_HTTP_Server_Raw		This application guides STM32Cube HAL API users to run an http server application based on the raw API of the LwIP TCP/IP stack.	-	-	MX
	LwIP_HTTP_Server_Socket_RTOS		This application guides STM32Cube HAL API users to run an http server application based on the socket API of the LwIP TCP/IP stack.	-	-	MX
	LwIP_TCP_Echo_Client		This application guides STM32Cube HAL API users to run a TCP echo client application based on a raw API of the LwIP TCP/IP stack.	-	-	MX
	LwIP_TCP_Echo_Server		This application guides STM32Cube HAL API users to run a TCP echo server application based on a raw API of the LwIP TCP/IP stack.	-	-	MX
	LwIP_TFTP_Server		This application guides STM32Cube HAL API users to run a tftp server application for STM32H7S7xx devices.	X	-	-
	LwIP_UDPTCP_Echo_Server_Netconn_RTOS		This application guides STM32Cube HAL API users to run UDP TCP echo server application based on the Netconn API of the LwIP TCP/IP stack.	-	-	MX
	LwIP_UDP_Echo_Client		This application guides STM32Cube HAL API users to run an UDP echo client application based on a raw API of the LwIP TCP/IP stack.	-	-	MX
	LwIP_UDP_Echo_Server		This application guides STM32Cube HAL API users to run a UDP echo server application based on a raw API of the LwIP TCP/IP stack.	-	-	MX
	RoT		OEMiRoT_Appli	This project provides an OEMiRoT boot-path application example. Boot is performed through the OEMiRoT boot path after authenticity and integrity checks of the project firmware.	X	-
		OEMiRoT_Boot	This project provides an OEMiRoT example. OEMiRoT boot path performs authenticity and integrity checks of the project firmware.	X	-	MX



Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Applications	RoT	STiRoT_Appli	This project provides a STiRoT boot-path application example. Boot is performed through the STiRoT boot path after authenticity and integrity checks of the project firmware.	X	-	MX
		STiRoT_iLoader	This project provides a STiRoT boot path immutable loader example. Boot is performed through the STiRoT boot path after write-protection check on the flash.	X	-	MX
	USB PD	USBPD_DRP_DRD	This application is an USBPD type C dual role power (DRP) and dual role data (DRD) application using FreeRTOS.	X	-	-
		USBPD_SNK	This application is a USB PD type C consumer using FreeRTOS™.	MX	-	MX
		USBPD_SRC	This application is a USB PD type C provider using FreeRTOS™.	MX	-	MX
	USB_Device	Audio_Standalone	This application is a part of the USB device library package using STM32Cube firmware. It describes how to use a USB device application based on the AUDIO class implementation of an audio streaming (out: speaker/headset) capability on the STM32H7Rx/7Sx devices.	MX	-	-
		CDC_Standalone	This application is a part of the USB device library package using STM32Cube firmware. It describes how to use a USB device application based on the device communication class (CDC) following the PSTN subprotocol in the NUCLEO-H7S3L8 devices using the OTG-USB and UART peripherals.	-	-	MX
		Composite_CDC_HID_Standalone	This application is a part of the USB device library package using STM32Cube firmware. It describes how to use a USB device application based on the CDC HID composite device on the NUCLEO-H7S3L8 device.	-	-	X
		HID_Standalone	This application provides an example of FreeRTOS™ on NUCLEO-H7S3L8 board. It shows how to develop a USB device human interface HID mouse-based application.	-	-	MX
		MSC_Standalone	This application is a part of the USB device library package using STM32Cube firmware. It describes how to use a USB device application based on the mass storage class (MSC) on the STM32H7Rx/7Sx devices.	MX	-	-
		Video_Standalone	This application is a part of the USB device library package using STM32Cube firmware. It describes how to use a USB device application based on the device video class in the NUCLEO-H7S3L8 devices using the OTG-USB peripherals.	-	-	X
	USB_Host	AUDIO_Standalone	This application is a part of the USB host library package using STM32Cube firmware. It describes how to use a USB host application. This application is based on the AUDIO class implementation of an audio streaming .wav files from an SD card capability on the STM32H7Rx/7Sx devices.	MX	-	-
		CDC_Standalone	This application is a part of the USB host library package using STM32Cube firmware. It describes how to use a USB host application based on the communication class (CDC) on the STM32H7Rx/7Sx devices.	MX	-	-
		DualClass_Standalone	This application is a part of the USB host library package using STM32Cube firmware.	X	-	-
		HID_RTOS	This application is an example implementation of a USB HID (human interface device) host with FreeRTOS™, supporting both high-speed (HS) and full-speed (FS) USB devices.	MX	-	-

Level	Module name	Project name	Description	STM32H7S78-DK	STM32H7RS_CUSTOM_HW	NUCLEO-H7S3L8
Applications	USB_Host	MSC_RTOS	This application is an example implementation of a USB MSC (mass storage class) host with FreeRTOS™, supporting high-speed (HS).		-	-
	Total number of applications: 41			19	0	22
Total number of projects: 287				61	3	223



Revision history

Table 2. Document revision history

Date	Version	Changes
12-Mar-2024	1	Initial release.
23-Jul-2024	2	Updated: <ul style="list-style-type: none">• Figure 1. STM32CubeH7RS firmware components• Table 1. STM32CubeH7RS firmware examples

Contents

1	Reference documents	2
2	STM32CubeH7RS examples.....	3
	Revision history	24
	List of tables	26

List of tables

Table 1.	STM32CubeH7RS firmware examples	5
Table 2.	Document revision history	24

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2024 STMicroelectronics – All rights reserved